Research Product 84-06

AUTOMATED TACTICAL SYMBOLOGY SYSTEM (TACSYM): SYSTEM DESIGN SPECIFICATIONS

Battlefield Information Systems
Technical Area
Systems Research Laboratory

March 1984

U.S. ARMY RESEARCH INSTITUTE for the BEHAVIORAL and SOCIAL SCIENCES

U. S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Technical Director

L. NEALE COSBY
Colonel, IN
Commander

Research accomplished under contract for the Department of the Army

PERCEPTRONICS, INC.

Technical review by

Beverly Knapp
Franklin Moses

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Research Product 84-06	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Automated Tactical Symbology System (TACSYM): System Design Specifications		5. TYPE OF REPORT & PERIOD COVERED Research Product 6. PERFORMING ORG. REPORT NUMBER PVASDS-1063-81
7. AUTHOR(s) Patrick Peck and Steven Johnston Perceptronics, Inc.		8. CONTRACT OR GRANT NUMBER(s) DAHC19-78-C-0018
9. PERFORMING ORGANIZATION NAME AND ADDRESS Perceptronics, Inc. 6271 Variel Avenue Woodland Hills, CA 91367		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2Q263739A793
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Research Institue 5001 Eisenhower Avenue Alexandria, VA 22333		12. REPORT DATE March 1984 13. NUMBER OF PAGES 96
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

L 20

18. SUPPLEMENTARY NOTES

Technically monitored by Dr. Franklin Moses and Ms. Beverly Knapp.
Other related ARI reports are RR 1369, TR 582, and RP 83-06.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Automated System Design
Tactical Symbols
Military Symbology
System Specifications
Graphic Portrayal
Symbology

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An automated catalog of tactical symbols (TACSYM) was developed to compare and evaluate existing and proposed symbologies for military applications. This document describes the system architecture - hardware and software configurations - and the overall system operating characteristics. The primary features of the automated tactical symbology system are: 1) database of over 700 symbols from 15 different military symbology sources; 2) interactive access allowing users to graphically display and print symbols from the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

database; 3) symbol building capability for updating database and experimenting with new symbols; 4) on-line tutorial explaining capabilities and characteristics of TACSYM; 5) query mechanisms allowing users to locate and display symbols by specific characteristics.

Research Product 84-06

AUTOMATED TACTICAL SYMBOLOGY SYSTEM (TACSYM): SYSTEM DESIGN SPECIFICATIONS

Patrick Peck and Steven Johnston
Perceptronics, Inc.

Submitted by
Franklin L. Moses, Chief
Battlefield Information Systems Technical Area

Approved as technically adequate and submitted for publication by
Jerrold M. Levine, Director
Systems Research Laboratory

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES
5001 Eisenhower Avenue, Alexandria, Virginia 22333

Office, Deputy Chief of Staff for Personnel
Department of the Army

March 1984

Army Project Number
2Q263739A793

Embedded Training and
Decision Support

ARI Research Reports and Technical Reports are intended for sponsors of R&D tasks and for other research and military agencies. Any findings ready for implementation at the time of publication are presented in the last part of the Brief. Upon completion of a major phase of the task, formal recommendations for official action normally are conveyed to appropriate military agencies by briefing or Disposition Form.

FOREWORD

The Battlefield Information Systems Technical Area of the Army Research Institute is concerned with the human resource demands of increasingly complex battlefield displays used to acquire, transmit, process, disseminate, and utilize information. Current research focuses on human performance problems related to the soldier system interface and is concerned with such areas as software development, the presentation of information on complex displays, user-oriented systems, decision making, systems integration, and utilization.

Of special interest are human factors problems related to developing and validating new ADP compatible symbology concepts for efficient display of tactically significant information. In a three-task symbology contract effort by Perceptronics, Inc., one task involved the development of an automated tactical symbology system (TACSYM). The TACSYM contains over 700 symbols from various military symbol sources. A user of TACSYM can index symbols by concept, category, or source and have these graphically displayed on a CRT terminal. The current document describes the overall architecture of the TACSYM system and its general operating characteristics. It is intended as a technical guidebook to assist those who will be deploying TACSYM on an existing computer installation.

EDGAR M. JOHNSON
Technical Director

AUTOMATED TACTICAL SYMBOLOGY SYSTEM (TACSYM): SYSTEM DESIGN SPECIFICATIONS

EXECUTIVE SUMMARY

Requirement:

To provide detailed documentation of the automated tactical symbology system (TACSYM) architecture and operations. TACSYM was developed as part one of a three task contract effort to investigate graphic (symbolic) portrayal of battlefield information. The catalog exists on magnetic tape for use on UNIX-like operating systems for state-of-the-art mainframe computers.

Procedure and Results:

Outlined in this system specification document are the overall and detailed system design of the TACSYM catalog. Following this, a detailed operational description is provided. Included are the hardware configurations for the original system, initial implementations description and functional capabilities. Interactive features, tutorial, database management, and file structure organization are also discussed. The final discussion provides detailed software system design logic.

Utilization:

The documentation of TACSYM provides an understanding of the structure, function, and operations of TACSYM from a programmer's perspective. It is intended as an aid to implementation of the software in future installations. Also, it provides the basis for the formation of guidelines for user interactions with a fielded version of TACSYM.

AUTOMATED TACTICAL SYMBOLOGY SYSTEM (TACSYM): SYSTEM DESIGN SPECIFICATIONS

CONTENTS

CONTENTS (Continued)

1. INTRODUCTION

1.1 Overview

The Automated Tactical Symbology System (TACSYM) was developed to catalogue, compare and evaluate existing and proposed military symbologies obtained from many sources. Its purpose is to produce symbology displays and hard copies for multiple symbol dimensions, such as military concept, category, symbol source, alpha-numeric information and discriminability.

The primary features of the Automated Tactical Symbology system are:

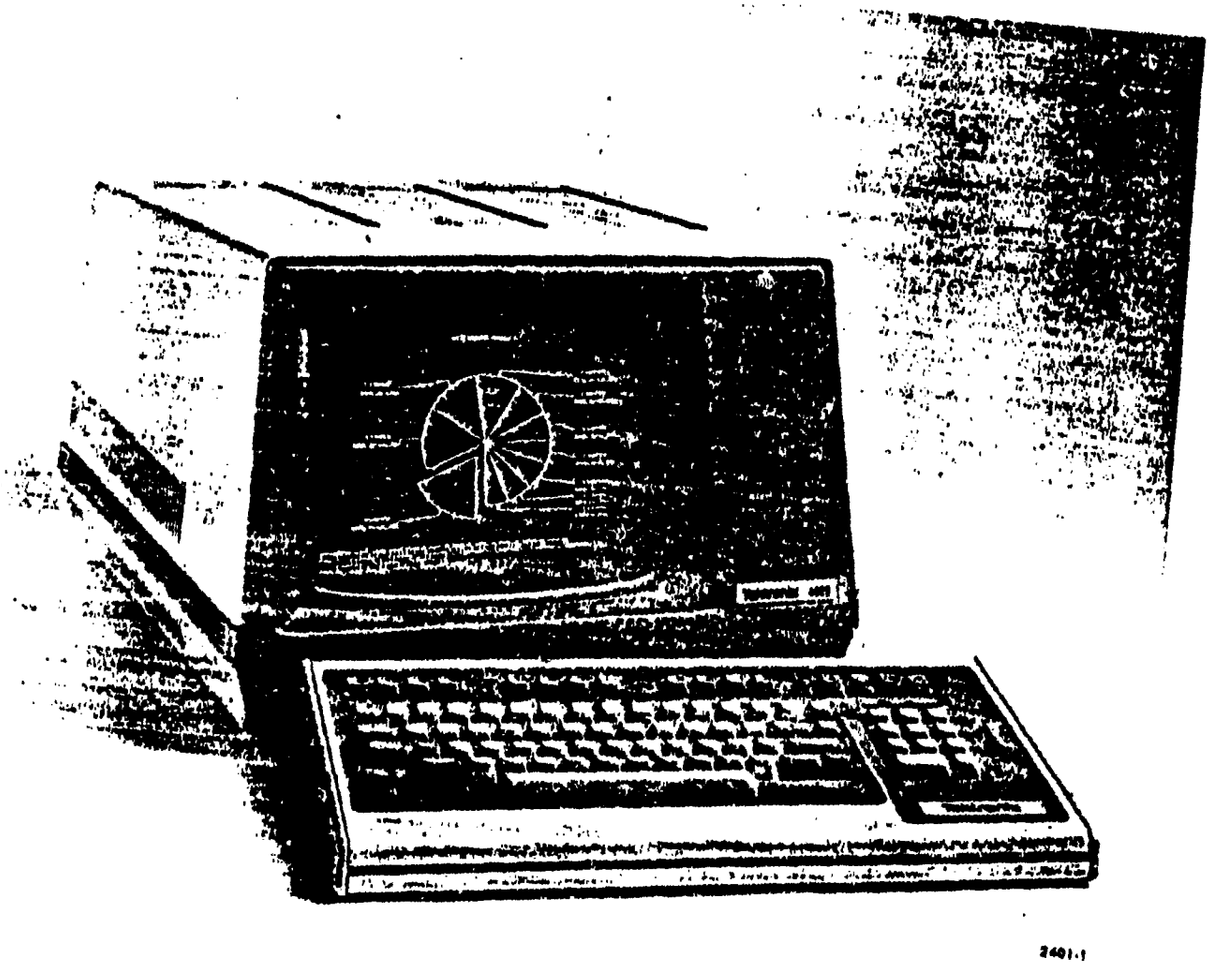
- Database of over 700 symbols from 15 different military symbology sources.
- Interactive access allowing users to graphically display and print symbols from the database.
- Symbol building capability for updating database and experimenting with new symbols.
- On-line tutorial explaining capabilities and characteristics of TACSYM.
- Query mechanisms allowing users to locate and display symbols by specific characteristics.

1.2 System Description

1.2.1 Hardware Configuration. TACSYM can use either the Tektronix 4025[1] or 4027 graphics terminal. These types of terminals have high resolution (640x482), and are black and white (4025) or color (4027). TACSYM currently uses only the black and white capability of the 4027.

---

[1]The use of a trademark does not constitute U.S. Army or Department of Defense endorsement of the product.



2401-1

Figure 1-1 4023 Computer Display Terminal

The hard copy device is the Tektronix 4631. This dry imaging device is connected directly to the Tektronix 4025. It can generate standard 21.59 x 27.94 cm camera ready copies of the Tektronix screen. Figure 1-2 is a picture of the Tektronix 4631.

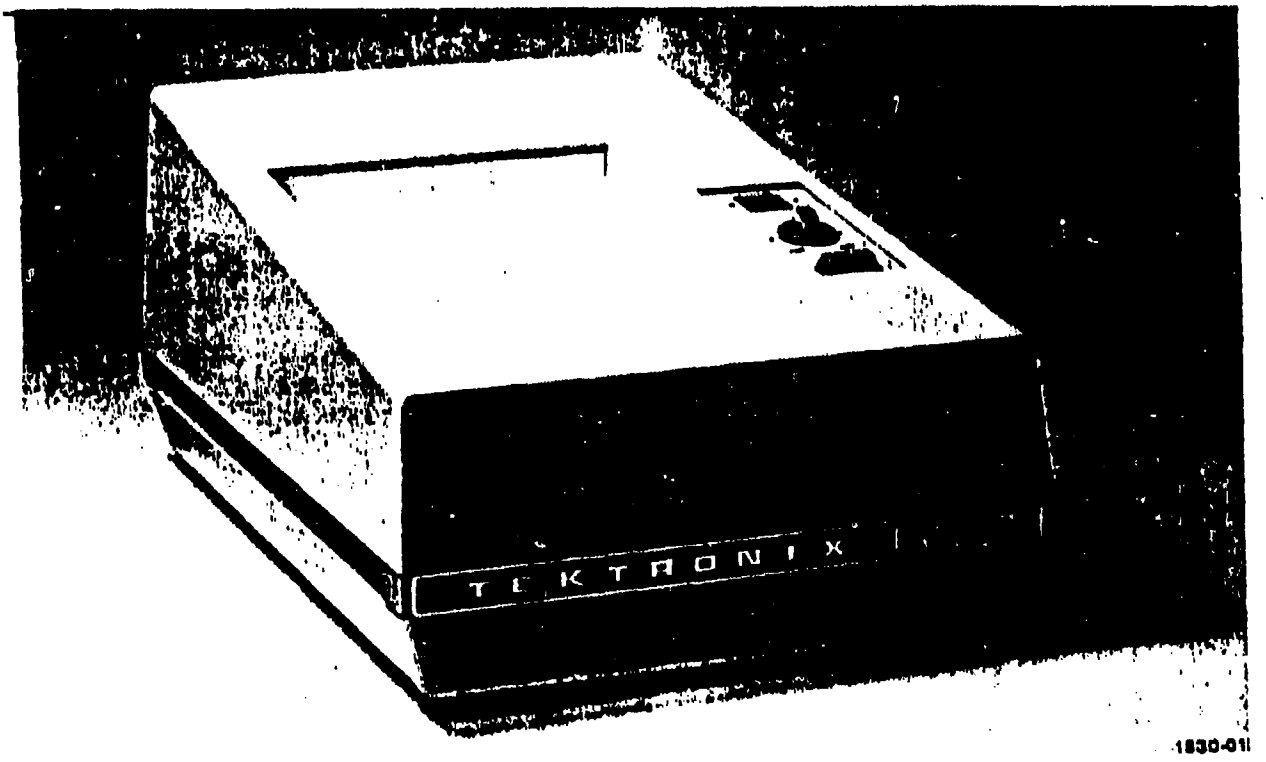


Figure 1-2 4631 Hard Copy Unit

TACSYM was implemented on a PDP 11/70 computer system which uses the UNIX operating system.

1.2.2 Implementation Description. TACSYM was written in highly structured modules using the C programming language. Structured programming provides for easy understanding and results in better programming practice. C is a

high level language (HLL) developed at Bell Laboratories originally for use in implementing operating systems, compilers, screen editors, syntactic parsers, and other utilities. However, in the past several years, C has become a popular applications language as well, providing computer scientists with an exacting control of data structures, file I/O string manipulation, and arithmetic operations not found in other HLL such as FORTRAN and PASCAL. In addition, the UNIX users group has advanced the state-of-the-art in portability of programming languages and operating systems (UNIX is written almost entirely in C). Existing software tools permit large software systems written in C to be transported easily across most new operating systems and new CPU's.

1.3 System Capabilities

1.3.1 Overview. Figure 1-3 represents a full system component view of TACSYM. The TACSYM illustration consists of tutorial and catalogue access components. The tutorial is an on-line, self help and information guide about the TACSYM catalogue. The catalogue component contains the symbol database and access, manipulation, and creation capabilities. The tutorial and catalogue access components are accessible via TACSYM's initial menu, and are easy to operate and understand.

1.3.2 Tutorial. The tutorial consists of areas of information that a TACSYM user can explore interactively on the system. The tutorial consists of a "System Operations" area and an "Executive Summary." Embedded self documentation such as on-line tutorials description of database access techniques, sample sets of embellished symbols and general rules for symbol construction are all contained in the tutorial section of TACSYM.

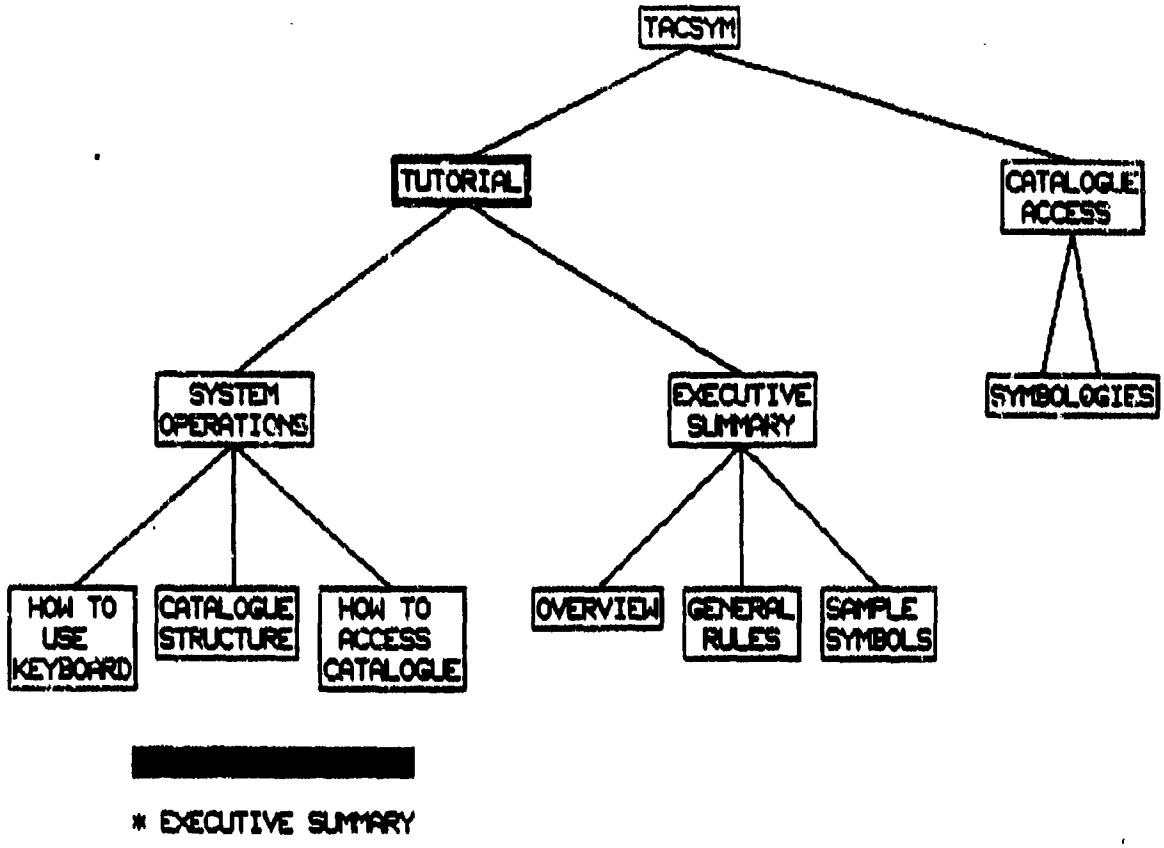


Figure 1-3 Heirarchical TACSYM Structure

1.3.3 Catalogue Access/Modification. TACSYM allows symbols and primitives to be defined as a collection of other symbols, primitives or graphical elements, generated via keyboard functions. Symbols are stored with detailed information including:

- o Category class (aviation, weapon, communication, etc.).
- o Concept name (helicopter, infantry, artillery, etc.).
- o Symbology source(s) (FM 21-30, Combat Power Symbology, NATO D-49, etc.).
- o Remarks (user information pertaining to the symbol).
- o Discriminability index for symbol comparison.

Categories, concepts and sources can be individually inserted provided they are non-existent in the database or deleted provided they are not represented

in any existing symbol. TACSYM allows for the removal of symbols and primitives by name only. The database can be searched on a number of different selection criteria such as:

- o Direct access of primitives, by a number associated with a name in the primitive list.
- o Direct access of symbols by number.
- o Simple enumerative display of all primitives.
- o Simple enumerative display of all symbols.
- o Simple enumerative display of all flagged symbols.
- o Display of all symbols for a selected source, e.g., FM 21-30.
- o Display all symbols for a selected concept.
- o Display all symbols for a selected category class.

TACSYM also provides the following modes of hard copy printing:

- o Direct user copy of screen content.
- o Automatic copy of all symbols in the database.

Figure 1-4 illustrates catalogue access/modification.

1.4 System Characteristics

The catalogue "categories" of TACSYM are military classifications which comprise a particular group of symbols. The symbols which are assigned to a category have some physical meaning in common with one another. The first two digits of a symbol number indicate its category.

Symbol Categories

1. Activity
2. Aviation
3. Communication
4. Vehicles
5. Tactical Unit
6. Measle
7. Combat Service Support
8. Nuclear Bio Chem
9. Obstacles and Fort
10. Installations
11. Weapon

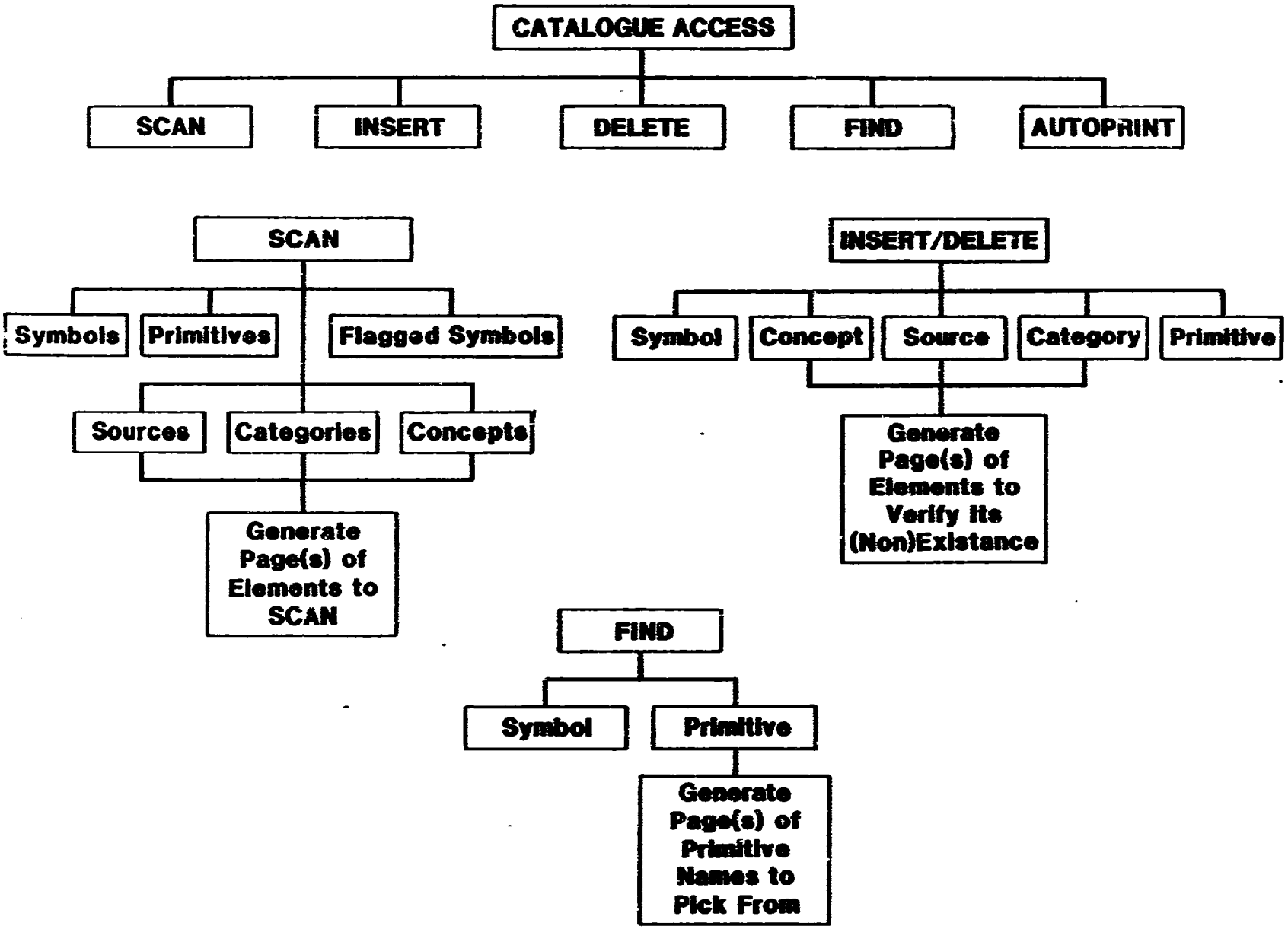


Figure 1-4 Catalog Access

The catalogue "sources" of TACSYM are names identifying groups of symbols which refer to a military symbology. The symbols of a particular source are common by the fact that they were designed to satisfy the requirements for communicating military information in some fashion.

Symbology Sources

1. Divras
2. NATO D-49 (1980)
3. TCO
4. FM 21-30
5. TOS
6. CPS
7. BETA Test Bed
8. MIFASS
9. TAOC-85
10. TACC
11. ITAOC
12. AF E-3A PPI
13. AF 407L/485 LT
14. PLRS
15. AFR 55-25
16. FM 101-5-1
17. CDEC-VIDS

The Catalogue Concepts of TACSYM are military descriptions about the symbology. These concepts further describe a symbology in relation to its graphics image and category (i.e., the concept fighter is used to describe an iconic of a plane contained in the Aviation category).

Symbol Concepts

1. Action
2. Bridging
3. Ferrying
4. Movement
5. Destroyed
6. Phoney
7. Propaganda
8. Responsibility
9. Snorkeling
10. Aerial
11. Antisubmarine
12. Aviation
13. Bomber
14. Close Air Support
15. Drone Aircraft
16. Fighter
17. Fixed Wing Aircraft
18. Helicopter
19. Marine
20. High Performance
21. Medevac
22. Rescue
23. Seaplane
24. Transport
25. Emit
26. Message Center
27. Signal
28. Nuclear
29. Formation
30. Reconnaissance
31. Jamming
32. Radar
33. Radio
34. Telephone
35. Teleprinter
36. Television
37. ADP Central
38. Elec. Navig. Aid
39. Microphones
40. Target Designator
41. Visual Station
42. Amphibious
43. Animal
44. Armoured
45. Boat
46. C-Cube
47. Cargo
48. Engine
49. Ferry
50. Air Cavalry

Symbol Concepts

51. Hovercraft
52. Operational
53. Communication
54. Over-snow
55. Personnel
56. Missile
57. Railway
58. Shooter
59. Sledge/sled
60. Vehicle
61. Ship
62. Airfield
63. Submarine
64. Tracked
65. Landing Site
66. Train
67. Wheeled
68. Unspecified
69. Landing Zone
70. Vehicles
71. Surface
72. Seaplane Station
73. Landing Vehicle
74. MICV
75. Air Defense
76. Ammunition
77. Air Mobile
78. Air Transportable
79. Air Naval Ground
80. Airborne
81. Antiaircraft
82. Antitank
83. Armour
84. Army Security Agcy
85. Artillery
86. Collecting Point
87. FA
88. Construction
89. CBR
90. Chemical
91. Combined Arms Army
92. CEWI
93. C2 Element
94. Decontamination
95. Command
96. Electronic
97. Electronic Warfare
98. Engineer
99. Food
100. Infantry

Symbol Concepts

101. Fuel
102. Irregular Forces
103. Maneuver Unit
104. Marines
105. Brigade
106. Military
107. Mortar Fire Unit
108. Motor Rifle
109. Mountain
110. Movers
111. Navy
112. Ordnance
113. Parachute
114. Reinforcement
115. Shooters
116. Special Forces
117. Bio or Chem Event
118. Contamination
119. Radioactive Area
120. Radioactive
121. Targets
122. Booby Trap
123. Bridge
124. Demolition
125. Fence
126. Data Proc. Unit
127. Dental
128. Maintenance
129. Major End Items
130. Medical
131. Hospital
132. Medical Supply
133. Mines
134. Missile Supply
135. Multi-Class
136. Multirole
137. Nuclear Storage
138. Personal Demand
139. Repair Parts
140. Wire
141. Obstacle
142. Subsistence
143. Traffic Control
144. Unknown Logistic
145. Water
146. Force
147. Headquarters
148. Landing
149. Logistics Unit
150. Marine Amphibious

Symbol Concepts	Symbol Concepts	Symbol Concepts
151. Aero Medical	161. Tank	171. Weapon
152. Motor Transport	162. Clothing	172. Smoke Generator
153. Petroleum Supply	163. Plane	173. Tracks
154. Shore Party Team	164. Flame Thrower	174. Vegetation
155. Support	165. Gun	175. Toxic Agent
156. Rear Area Operations	166. Mortar	176. Trenches
157. Service	167. Naval Gunfire	177. Zone
158. Trains	168. Rifle	178. Commo Site
159. Ground Attack	169. Rocket	179. Howitzer
160. ICV	170. Rocket Launcher	

2. OVERALL DESIGN

2.1 System Organization

Figure 2-1 represents the logical structure of TACSYM. TACSYM consists of two components: Tutorial and Query. During execution, each component runs with dependency on the database and the software support system. The introduction processor, via user command, determines which direction to take in TACSYM. Figure 2-2 is a table of source files that are compiled and linked into executable modules for Tutorial and Query. The detailed description of these source files appears in Section 3-1, overview of software modules.

2.1.1 Introduction Processor Component. The introduction processor component generates the first page of the TACSYM catalogue. It provides three alternatives to the user. One is to examine the tutorial describing the catalogue and its operations. Two involves the-top menu of the catalogue access and modification section. Three terminates TACSYM and erases the screen. The system file INTRO1.C contains the main entry point for TACSYM. This file controls the execution of the Tutorial and Query components of the system. It also controls the termination process of TACSYM. The tutorial is executed by a procedure call. The query component is accessed by the UNIX "execl" command. Since the query component is a separate program (DATABASE), the "execl" command is used to call it. Upon exit from the main procedure of the query component, TACSYM is called again via the "execl" command. TACSYM and DATABASE, although containing many of the same routines, are two separate executable programs each smaller than the 64K byte maximum process size for most PDP 11's.

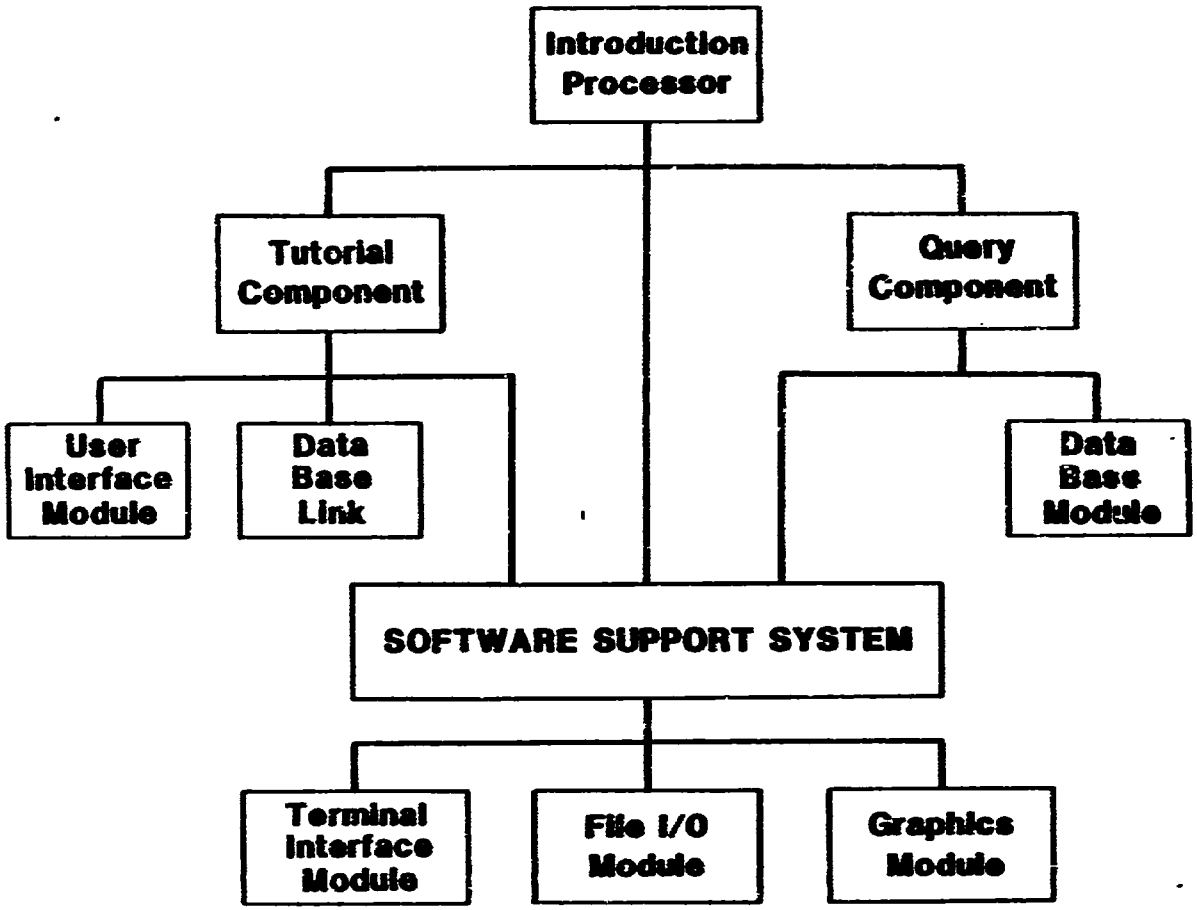


Figure 2-1 Logical Structure of Tacsym Software

Tutorial Source Files	Query Source Files
arc1.c	arc1.c
arc2.c	arc2.c
graphic.c	database.c
intro1.c	graphic.c
intro2.c	rawcook.c
introduction.c	scan.c
rawcook.c	str.c
str.c	tek1.c
tek1.c	trig.c
trig.c	

The source files contain the routines needed to operate and process TACSYM.

The source files contain detailed information which includes:

- o VARIABLE DEFINITIONS (GLOBAL AND LOCAL)
- o PARAMETER Definitions
- o Detailed explanations of the ROUTINE LOGIC
- o Detailed explanations of the SOURCE CODE

Figure 2-2

2.1.2 Tutorial Processor Component. The tutorial processor component is a system teaching mechanism. It allows users to learn all the capabilities of the system. By paging through the tutorial, users can read about and practice the functions of the system. They can also learn about the meanings of symbols, how symbols are composed, how the catalogue access mechanism works as well as the history of the TACSYM project.

The file INTRODUCTION.C is a command file processor with special characteristics. It processes text files which contain embedded commands. The commands range from controlling text pages to identifying optional branching sequences in the tutorial hierarchy to generating graphics on the Tektronix. In order to generate graphics, the graphics related files GRAPHIC.C, ARC1.C, ARC2.C, TEK1.C are linked together with INTRODUCTION.C. In order to perform the query program functions in INTRO2.C, it is linked to INTRODUCTION.C.

2.1.3 Query Processor Component. The query processor component is the program which accesses and modifies the database. The system file DATABASE.C contains the main entry point for the query processor. The menu driven access to the database is controlled by DATABASE.C and SCAN.C respectively. SCAN.C contains procedures used to scan information in the database. DATABASE.C also contains procedures which allow for modification of the database. Insertion and removal of data from the database is controlled by DATABASE.C.

2.1.4 Software Support System. The software support system used by TACSYM consists of the system files ARC1.C, ARC2.C, GRAPHIC.C, RAWCOOK.C, STR.C,

TEK1.C, and TRIG.C. The software support system is used by the introduction processor, tutorial component and query component. The software support system was designed so that any component of TACSYM could use its highly structured and diversified routines. String manipulation, structured I/O, and low level data manipulation routines are used to interface TACSYM with the Tektronix 4025 graphics terminal and database file I/O. The graphic procedures used by the query and database components are complex mathematical routines used throughout TACSYM in generating symbols, primitives and graphical elements. The routines of the software support system utilize the capabilities of the operating system, the C programming language, and the Tektronix 4025 graphic terminal.

2.1.5 User Interface. The user interface interprets and responds to user entered commands. The 12 function keys of the Tektronix 4025 have been programmed to perform various functions. Other keys on the keyboard have been programmed as well for specific operations. Figure 2-3 is a layout of the TACSYM keyboard. Table 2-2 lists the use of the function keys. TACSYM always lets users know the keyboard options they have throughout its operation. Keyboard overlays are provided for use with TACSYM.

2.2 Database Organization

The TACSYM database utilizes the hierarchical file system structure provided by the UNIX operating system. The file system is tree structured permitting construction of directories of sub-trees. This type of file system organization permits a well structured database.

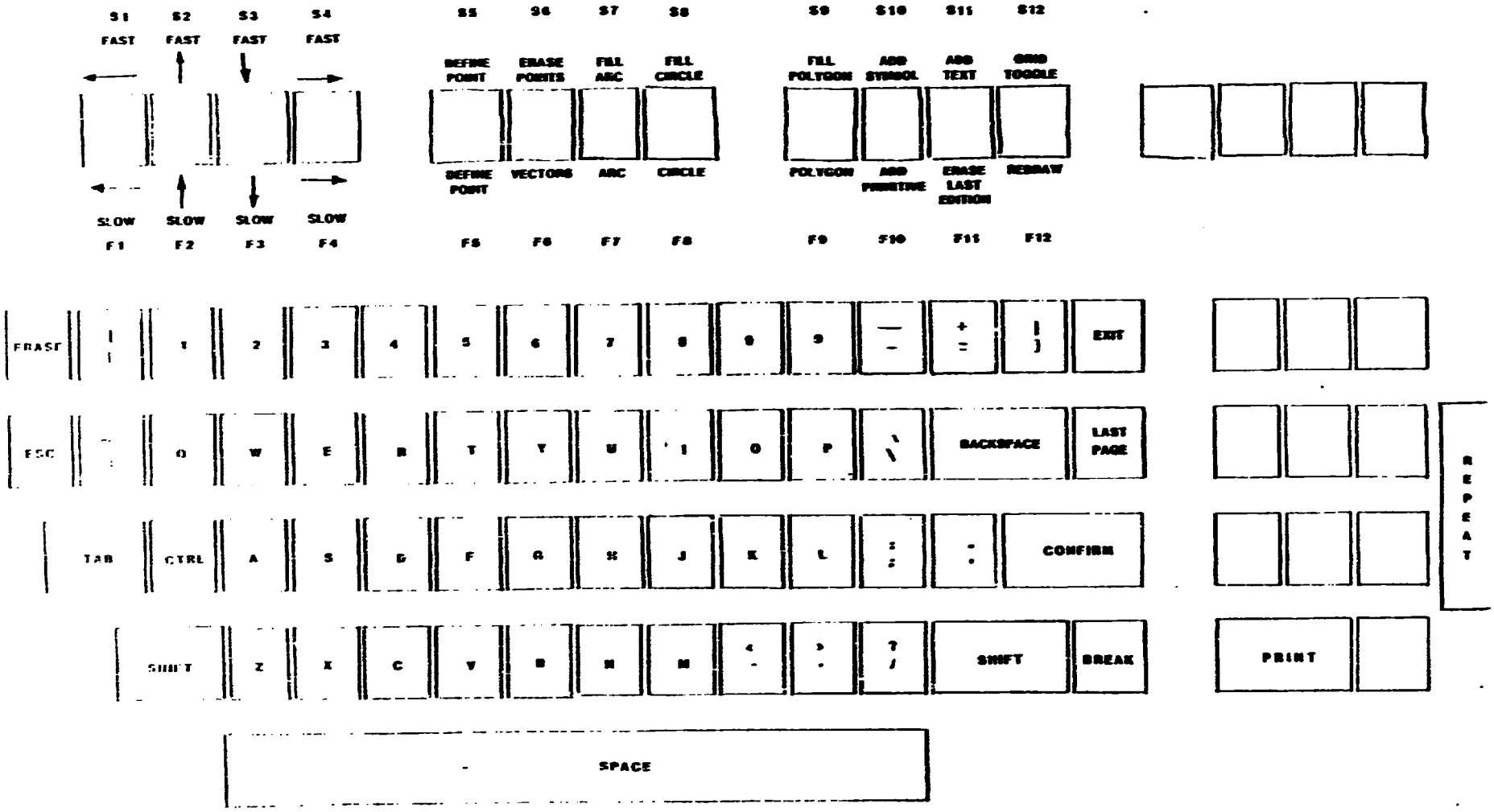


Figure 2-3 TACSYM Keyboard

KEYBOARD

S1:	Moves graphics cursor 5 positions to the left.
F1:	Moves graphics cursor 1 position to the left.
S2:	Moves graphics cursor 5 positions up.
F2:	Moves graphics cursor 1 position up.
S3:	Moves graphics cursor 5 positions down.
F3:	Moves graphics cursor 1 position down.
S4:	Moves graphics cursor 5 positions to the right.
F4:	Moves graphics cursor 1 position to the right.
S5:	Defines a point to be used in creating primitives of symbols, places an X at the defined point.
F5:	Same function as S5.
S6:	Erases all defined points and positions the cursor at the first point it originally defined.
F6:	Connects any points currently defined by vectors. Maximum of five vectors will be drawn since six (6) points is the maximum allowed to be defined, at one time.
S7:	Fills an arc defined by last three points defined. The base of the arc is defined by the first and third points.
F7:	Draws an arc defined by last three points defined. The ends consist of the first and third points defined.

Table 2-2 Function Key Description

S8:	Draws a filled circle which is defined by last two points defined given any rotation.
F8:	Draws a circle which is defined by last two points defined given any rotation.
S9:	Fills a polygon, which can have at most six sides but no side is allowed to be intersected by other sides. Maximum of six points can be used to define the polygon with the last point defined automatically connected to the first point defined. A minimum of three points is required.
F9:	Draws a polygon, which can have at most six sides. Maximum of six points can be used to define the poygon, with the last point defined automatically connected to the first point defined. A minimum of three points is required.
S10:	Add symbol asks for a symbol number and inserts that symbol relative to where the cursor is currently at.
F10:	Add primitive asks for a primitive name and inserts that primitive relative to where the cursor is currently at.
S11:	Add text allows the insertion of any keyboard text. It is terminated by pressing any function key or when there is no room on the right or bottom side of the graphics window. When there is no room on the right side, the cursor is positioned on the next line, in line where text insertion started on the previous line. In the case where no room is left on the bottom line the cursor is positioned at the top line, in line where text insertion started on the bottom line.

Table 2-2 Continued

F11:	Erases last addition clears and redraws all work performed in the graphics window except for last item inserted.
S12:	Grid toggle is a toggle switch which toggles on and off grid liner and the boundry coordinates of the graphics window.
F12:	Redraw clears and redraws everything in the graphcis window.
EXIT:	Used to back out one level of the program upon each press.
CONFIRM:	Used in completing an input string, paging forward, or in some cases exiting. Treated as a carriage return.
BACKSPACE:	When typing text used to backspace.
PREVIOUS PAGE:	In tutorial the previous page in a sequence will be generated.
REPEAT:	In tutorial the current page will be generated again.
PRINT:	If the hard copy device is on and connected, a hard copy of the current page of the Tektronix's will be generated.

Table 2-2 Continued

2.2.1 Data Base Management System. The DBMS combines features of the UNIX operating system and file I/O processing capabilities of the C programming language. TACSYM utilized the functionability of C and UNIX in operations such as sorting, file I/O, updates, deletions and other database maintenance operations. The concepts of data integrity, data access and limited redundancy are manifest in TACSYM. The DBMS for TACSYM is classified as a self-contained system indicating that queries on the database are fixed or predefined. The attractiveness of TACSYM is embedded self-documentation and helpful menu commands which make it easy to use. The menus permit easy data manipulation.

2.2.2 Database. Figure 2-4 illustrates the TACSYM DATABASE implementation under UNIX. The Control Directory contains the following TACSYM Control Files --

- o TACSYM - Executable load module
- o Symbol - List of symbol numbers in the database
- o Flagged Symbol - List of flagged symbol numbers in the database
- o Source - List of symbology source names
- o Concept - List of symbology concept names
- o Category - List of symbology category names
- o Primitive - List of unique primitive names used to construct symbols

and the following sub-directories --

- o Symbols - Contains a pair of files for every symbol inserted into the database. The n.n.n file of a symbol contains that symbol's graphical description and the Tn.n.n. file contains a symbols text description. The range of n.n.n is 0.0.0. to 99.999.999, allowing the database to contain 10 million symbols (if any file system could support that many).

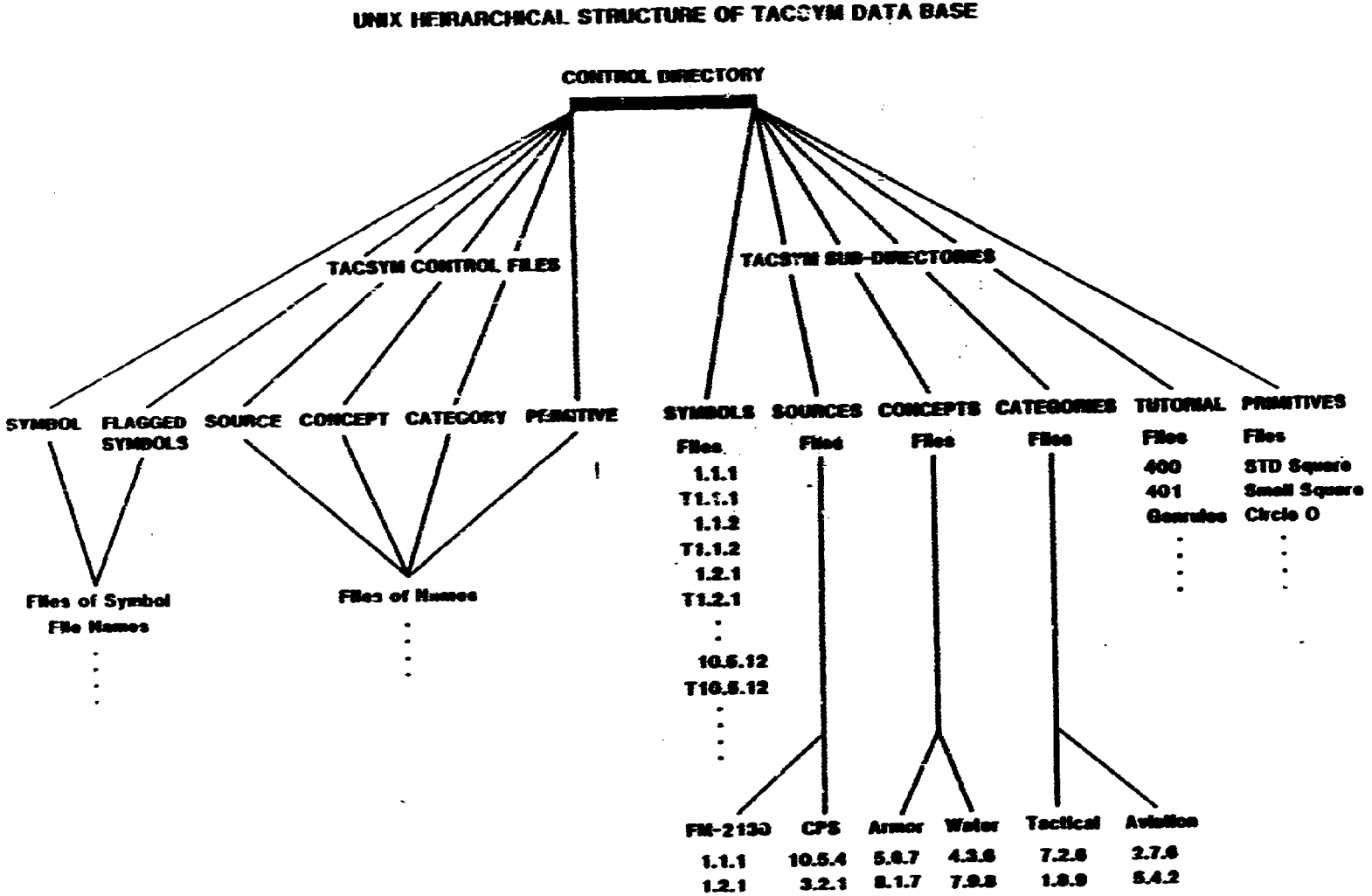

UNIX HEIRARCHICAL STRUCTURE OF TACSYM DATA BASE
CONTROL DIRECTORY
TACSYM CONTROL FILES
TACSYM SUB-DIRECTORIES
SYMBOL
FLAGGED SYMBOLS
SOURCE
CONCEPT
CATEGORY
PRIMITIVE
SYMBOLS
SOURCES
CONCEPTS
CATEGORIES
TUTORIAL
PRIMITIVES
Files of Symbol File Names
Files of Names
Files
1.1.1
T1.1.1
1.1.2
T1.1.2
1.2.1
T1.2.1
10.5.12
T10.5.12
Files
Files
400
401
Circle O
FM-2150
CPS
Armor
Water
Tactical
Aviation
1.1.1
1.2.1
10.5.4
3.2.1
5.6.7
8.1.7
4.3.6
7.9.8
7.2.6
1.8.9
2.7.6
5.4.2

Figure 2-4

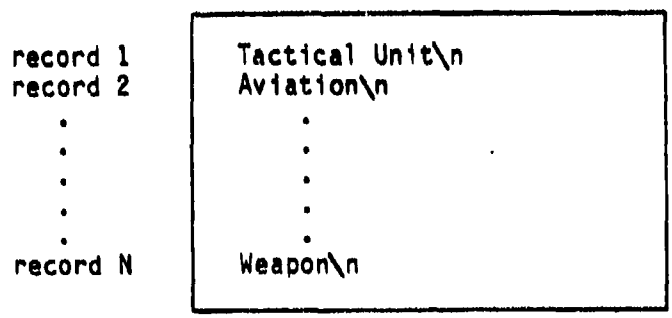
- o Sources - Contains files for each symbology source in the database. Each source file contains the list of symbol numbers in that source.
- o Concepts - Contains files for each symbology concept in the database and each file contains a list of symbol numbers representing the concept.
- o Categories - Contains files for each symbology category (number of categories is small -- approx. 11). Each category file contains a list of symbol numbers in the category.
- o Primitives - Contains files of primitive basic shapes and widely used symbol parts. When constructing symbols the primitives can be accessed by name and used as building blocks. Each file contains graphical descriptors necessary to construct the shape.
- o Tutorial - Contains files of text and graphical descriptions. These files are opened and processed by the file processor program used during the tutorial part of TACSYM.

2.3 File Structures

2.3.1 Catalogue Database.

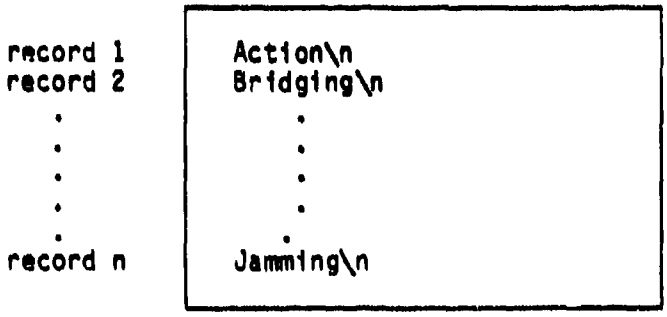
Category Control File

The category control file is of variable length and each record can contain no more than twenty ASCII characters. A newline character is appended at the end of each record.



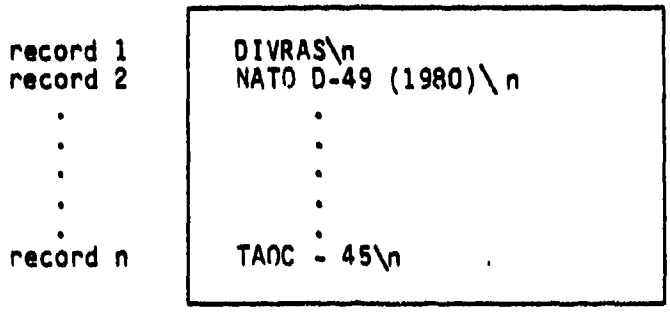
Concept Control File

The concept control file is of variable length and each record can contain no more than twenty ASCII characters. A newline character is appended at the end of each record.



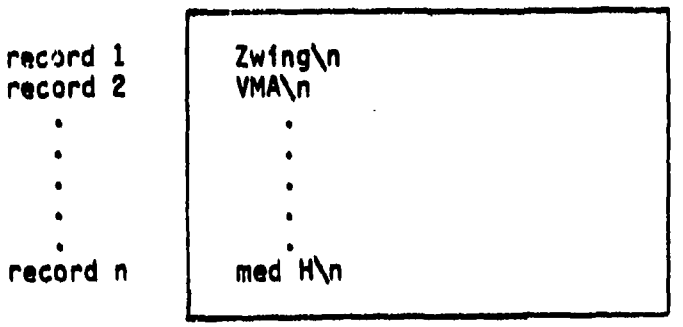
Source Control File

The source control file is of variable length and each record can contain no more than fifteen ASCII characters. A newline character is appended at the end of each record.



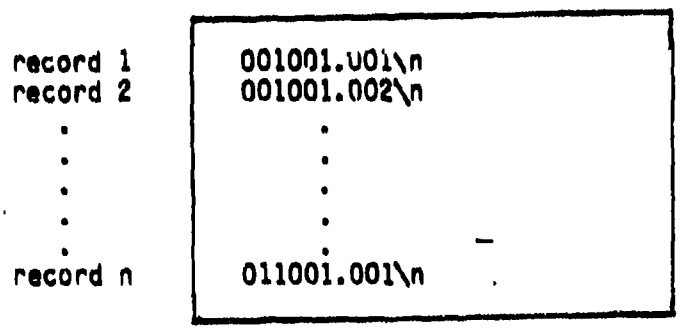
Primitive Control File

The primitive control file is of variable length and each record can contain no more than fifteen ASCII characters. A newline character is appended at the end of each record.



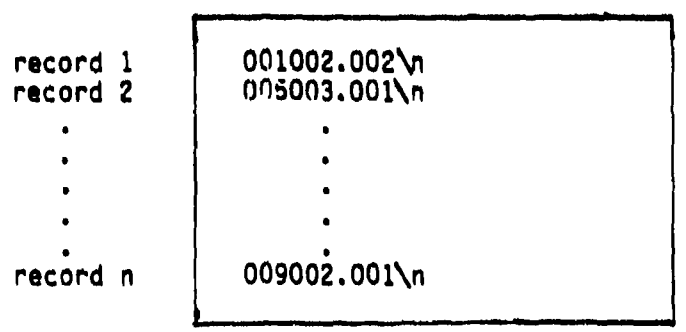
Symbol Control File

The symbol control file is of variable length and each record contains ten ASCII characters. A newline character is appended at the end of each record.



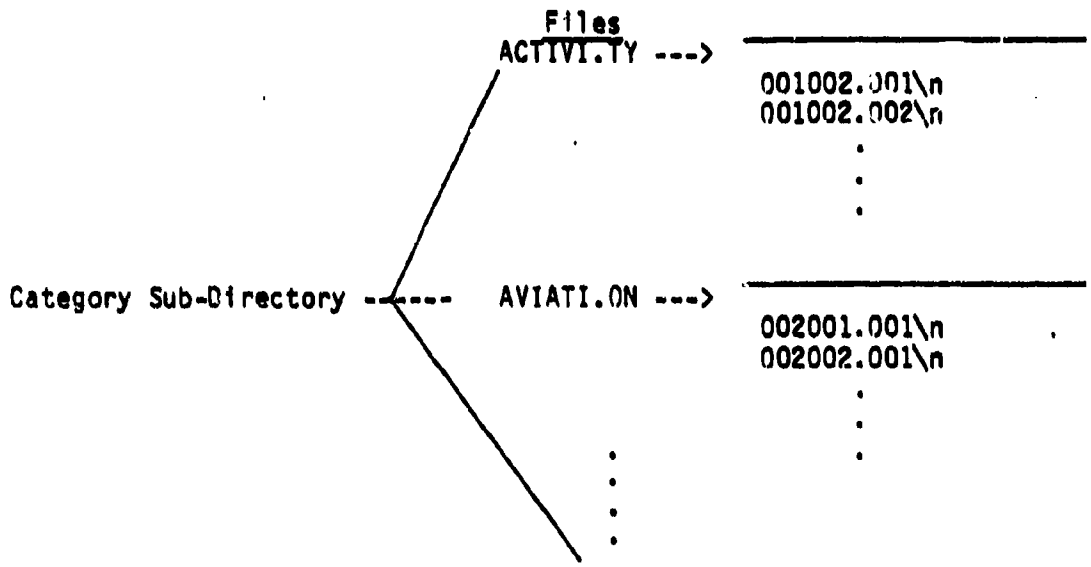
Flagged Symbol Control File

The flagged symbol control file is of variable length and each record contains ten ASCII characters. A newline character is appended at the end of each record.



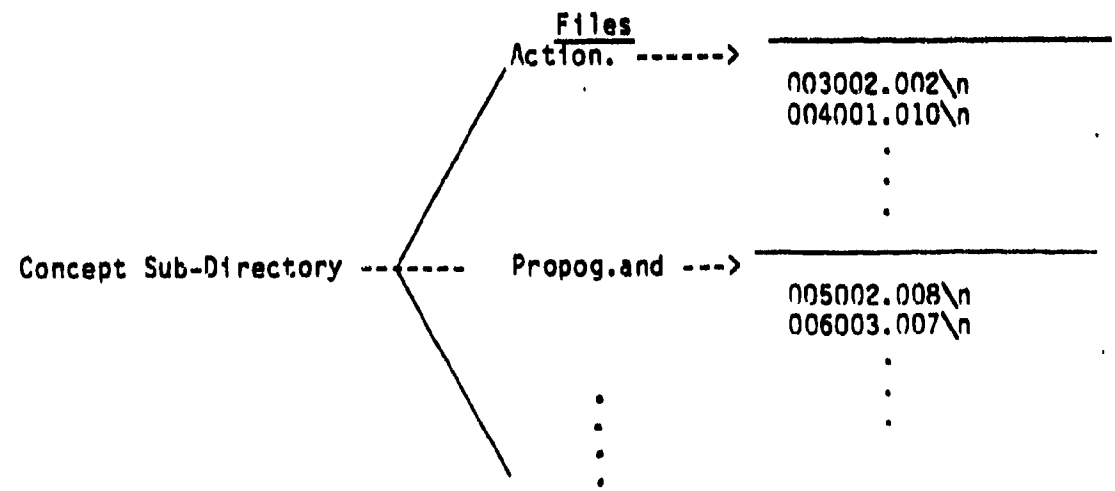
Category Sub-Directory

The category sub-directory contains files corresponding to each category. These contain record lengths of ten ASCII characters with a newline character appended to each record and are variable length files.



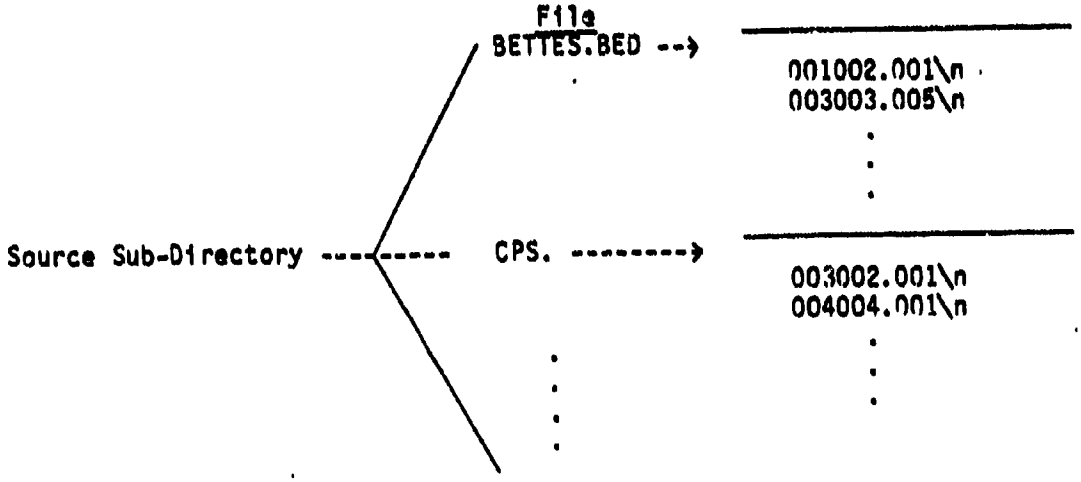
Concept Sub-Directory

The concept sub-directory contains files corresponding to each category. These files contain record lengths of ten ASCII characters with a newline character appended to each record and are variable length files.



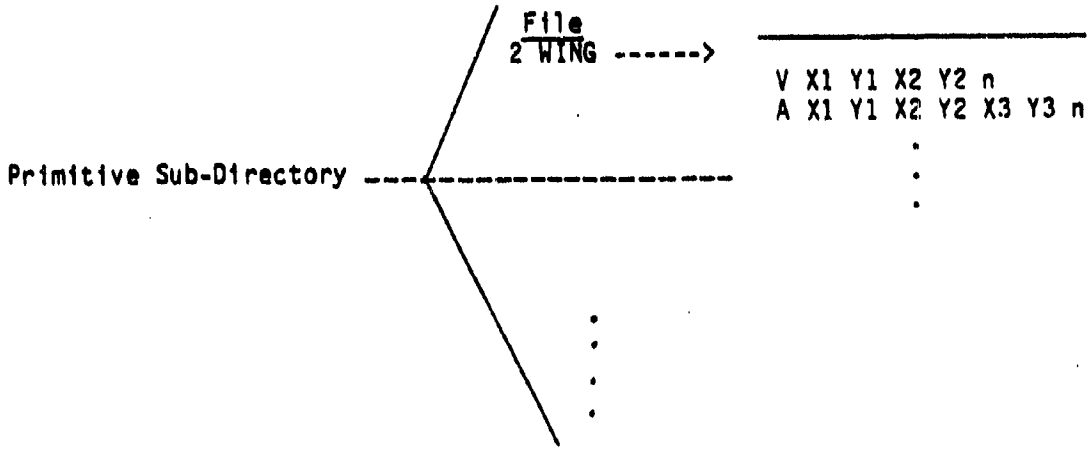
Source Sub-Directory

The source sub-directory contains files corresponding to each source. These files contain records of ASCII characters with a newline character appended to each record and are variable length files.



Primitive Sub-Directory

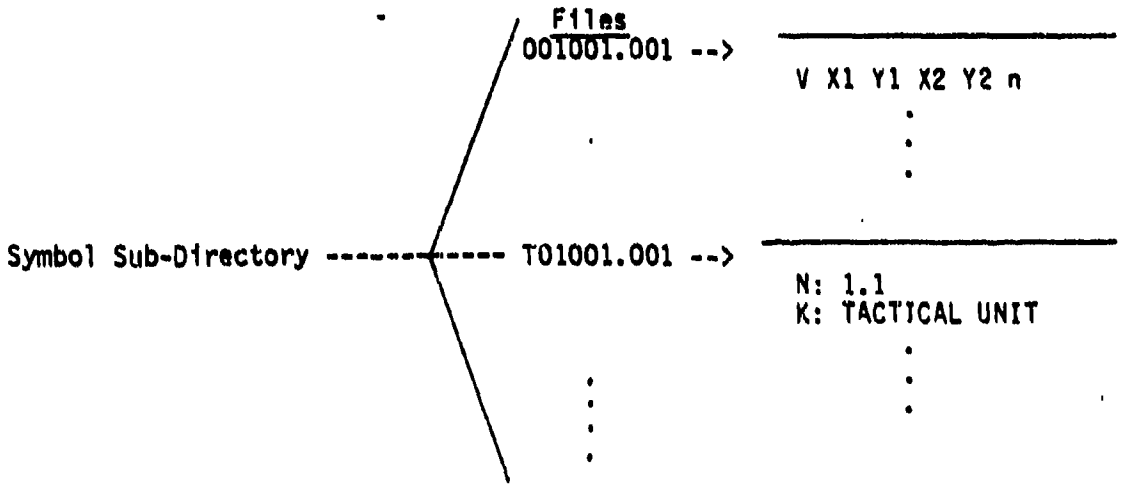
The primitive sub-directory contains files corresponding to each primitive. These files are variable length and have variable length records.



The file contents are explained further in section 2.4.

Symbol Sub-Directory

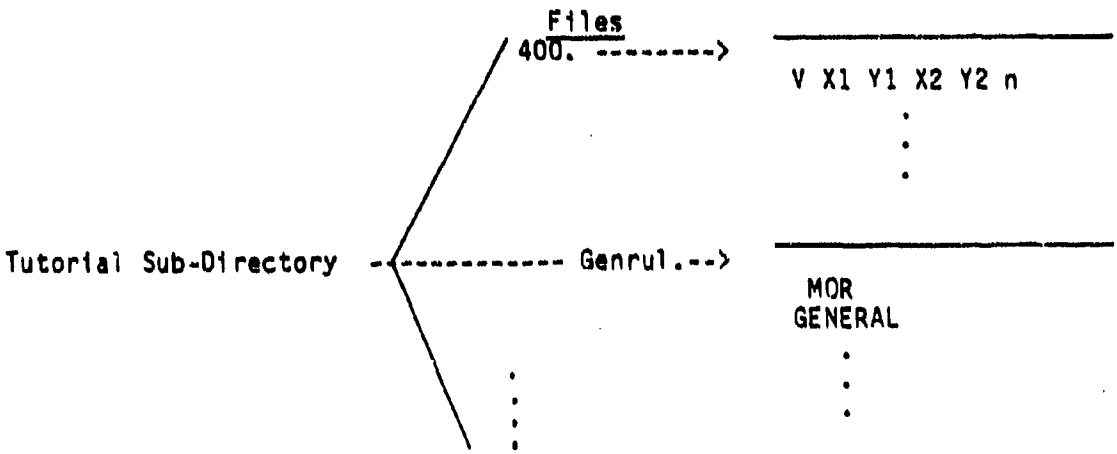
The symbol sub-directory contains files corresponding to each symbol and to the symbols text information. These files are variable length and have variable length records.



The file contents are explained further in section 2.4.

Tutorial Sub-Directory

The tutorial sub-directory contains files corresponding to symbols, primitives, text and menus that are used in the tutorial section of TACSYM.



The file contents are explained further in section 2.4.

2.4 Data Structure Definition

In section 2.3.1 the primitive, symbol and tutorial sub-directory files were introduced. They are explained further here. Figure 2-5 is a conceptual view of how the graphic elements, primitives and symbols make up a symbol representation.

Graphic Elements

Graphical elements consist of vectors, arcs, filled arcs, circles, filled circles, polygons, filled polygons and text. These elements can be "owned" by symbols or primitives. They are represented in the files as follows:

Description	Identifier	Defined By
Vectors	'v'	up to 6 pairs of x,y coordinates
Arcs	'a'	2 pairs of x,y, coordinates
Filled Arcs	'b'	" "
Circle	'c'	" "
Filled Circle	'd'	" "
Polygon	'g'	3-6 pairs of x,y, coordinates
Filled Polygon	'h'	" "
Text String	't'	1 pair of x,y coordinates followed by a text string

Primitives

Primitives can be composed of (1) symbols, (2) other primitives, (3) graphical elements, and (4) any combination of 1, 2, and 3. Primitives are identified by a unique file name. When contained in other symbols or primitives they are identified as:

'P' ∅ NAME ∅ X1 ∅ Y1\n

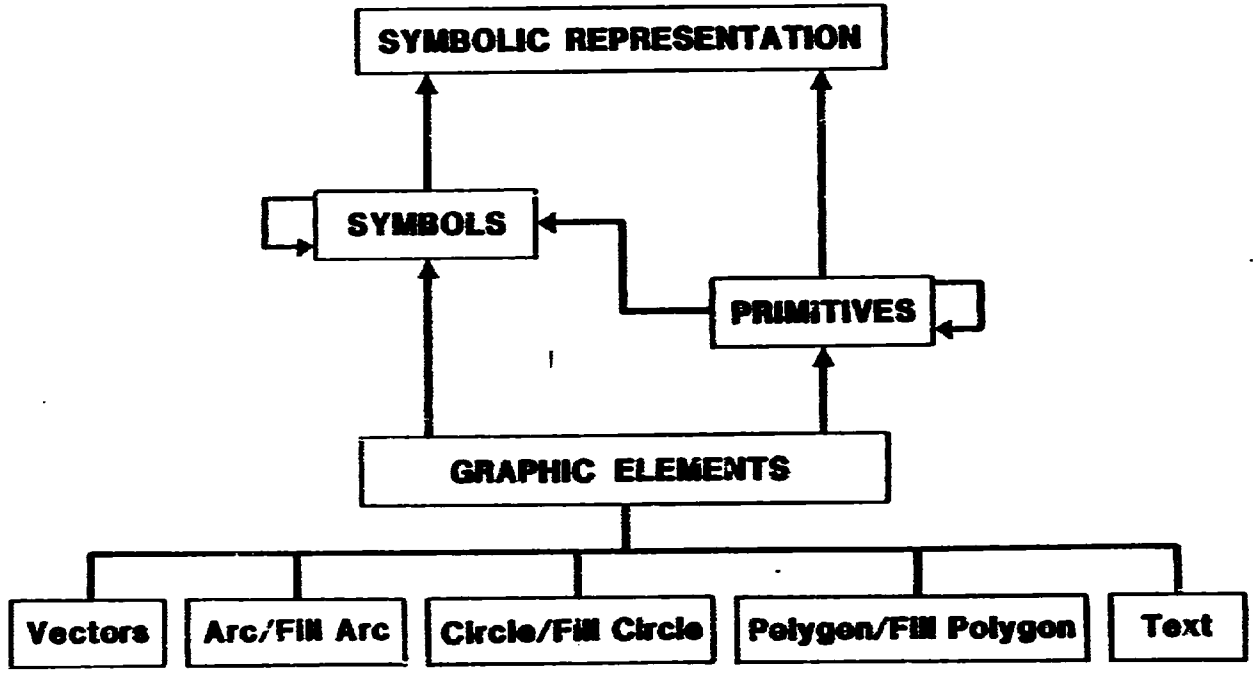

SYMBOLIC REPRESENTATION
SYMBOLS
PRIMITIVES
GRAPHIC ELEMENTS
Vectors
Arc/Fill Arc
Circle/Fill Circle
Polygon/Fill Polygon
Text

Figure 2-5

Symbols

Symbols can be composed of (1) other symbols, (2) primitives, (3) graphical elements, and (4) any combination of 1, 2, and 3. Symbols are identified by a unique file name, with the format of n.n.n. When contained in other symbols or primitives they are identified as:

'S' ∅ nnnnnn.nnnb X1 ∅ Y1\n

For every symbol in the database there are two files defining it:

nnnnnn.nnn - contains the graphic elements defining the symbol.

tnnnnn.nnn - contains the text and control information pertaining to the symbol.

The text file contains the text information about the symbol. As the text is keyed in when creating the symbol, it is written to this file. A text file consists of the symbol number, category, primary concept, secondary concept, sources, remarks and a flag indicator. The format for a text file is:

```
N :  n.n.n\n
K :  CATEGORY/\n
C :  PRIMARY CONCEPT/\n
1st LINE of SECONDARY CONCEPT\n
2nd LINE of SECONDARY CONCEPT\n
S :  1st LINE of SOURCES\n
2nd LINE of SOURCES\n
R :  1st LINE of REMARKS\n
2nd LINE of REMARKS\n
F :  FLAG (y or n)
```

The length of this file is variable and each record's length except for the flag indicator record is variable. The name of the text file is determined

by a T substituted for the first digit of the symbol's file name (i.e., symbol 007008.001 text file would be T07008.001).

Tutorial Text Files

The tutorial text files are composed of text with special commands preceded by a slash. These commands communicate to the program such things as draw any graphical element at a specific location on the monitor, erase the monitor, wait for a user reply, generate a symbol at some location and many other functions. These commands are defined and listed in section 3.2.

3. DETAILED SYSTEM DESIGN (SOFTWARE)

3.1 Overview of Software Modules

The software logical components of TACSYM include the Introduction processor, Tutorial processor, Query processor and the Software Support System. The system files which comprise these modules are listed and briefly described in Table 3-1. The files are designed so they can be used across different modules. Table 3-2 groups the files required by system function. The remaining sections in this chapter provide a detailed synopsis and description of all the software procedures in TACSYM.

3.2 Introduction Processor Component

intro1.c

NAME
MAIN - Main procedure for TACSYM program.

SYNOPSIS
MAIN ()

DESCRIPTION
Main, generates the initial page of the TACSYM catalogue and controls the system options available to the user.

arc1.c	-	An extension of graphic.c, arc1.c contains the subroutine arc used in computing and displaying arcs or filled arcs.
arc2.c	-	An extension of arc1.c, arc2.c contains the subroutine recur called by the arc subroutine. Recur is called repeatedly until criteria about the arc being computed are met.
database.c	-	Controls database access and manipulation of symbols.
direct.h	-	Header file containing defined constants used by more than 1 file.
graphic.c	-	Contains the subroutines used in generating symbols, primitives and graphical elements.
intro1.c	-	Controls execution of TACSYM's tutorial and database access components.
intro2.c	-	An extension of introduction.c used in linking to database routines which are used in the tutorial.
introduction.c	-	Controls the tutorial mechanism of processing text files with embedded commands.
rawcook.c	-	Sets terminal to echo keyboard input on the monitor or turns echo capability off.
scan.c	-	An extension of database.c, scan.c controls all scanning procedures.
str.c	-	File I/O subroutines used in supporting data manipulation from the terminal and the database files.
tek1.c	-	Subroutines used in communicating between the Tektronix graphics system and the software components.
trig.c	-	Trigonometric subroutines used in supporting graphical calculations.

TABLE 3-1

INTRODUCTION	TUTORIAL	DATABASE ACCESS	SOFTWARE SUPPORT SYSTEM
intro1.c	introduction.c intro2.c	database.c scan.c	arc1.c arc2.c graphic.c rawcook.c str.c tek1.c trig.c

TABLE 3-2

3.3 Tutorial Processor Component

INTRODUCTION.C

NAME
TUTOR - Main controller routine for the command file processor used in TACSYM's tutorial.

SYNOPSIS
TUTOR ()

DESCRIPTION
Tutor is the main driver for the tutorial component of tacsym. Initially it sets up the tektronix screen and generates all the menus that will be needed in the tutorial. It loads the main text file "TEXTFILE into the options structure. It calls LEARN1 which defines the namefunction keys that will be needed in the tutorial. The procedure SETUP is called within a loop. The returned value from SETUP controls the program.

NAME
SETUP - Main file I/O routine

SYNOPSIS
SETUP (K)
K - Indice of the options structure, which contains a text file to
open and process.

DESCRIPTION
SETUP, opens the file whose name is stored in the options structure. It processes the file a character at a time. All file commands are preceded by a backslash \ . The routine CHECK is executed when a command is encountered. If the option flag is 1, LOAD is executed to process the user controlled option's list. SETUP returns the index of the options structure which contains the next file to process. If -1 is returned then the program is terminated.

INTRODUCTION.C

NAME
LOAD - Controls the options menu which is user selectable.

SYNOPSIS
LOAD (K)
K - Index of option structure where new file will be added (K+1) or existing one deleted (K-1).

DESCRIPTION
LOAD, scans the option list which was generated in SETUP. Scanning is done by highlighting the menu items. When confirm is pressed, the information for the option selected is moved into the K+1 space in the options structure. If exit is pressed k-1 is returned.

NAME
CHECK - Processes a command in a file.

SYNOPSIS
CHECK (i)
i - Current index of options structure where the current file to be process resides.

DESCRIPTION
CHECK reads in a command, validates it, sets the appropriate flags, reads in any information that might be contained on the command line and via a switch statement processes the command.

INTRODUCTION.C

NAME
SW - Controls cursor movement in the graphic area during the tutorial.

SYNOPSIS
SW (a)
a - Integer value corresponding to one of 8 cursor movement options.

DESCRIPTION
SW, controls the cursor movement in the graphics window during the tutorial on the function keys.

NAME
MORE - Controls paging mechanism.

SYNOPSIS
MORE (ch, i)
ch - Integer character to indicate which paging option to perform.
i - Index of current file being processed in option structure.

DESCRIPTION
MORE, controls the paging mechanism's of generating a previous page, repeating the current page or going to the next page.

NAME
DONE - Displays blank menu for the "more" command.

SYNOPSIS
DONE ()

DESCRIPTION
DONE, displays the blank menu and sets the exit flag to 1. It's used in conjunction with the "more" command.

INTRODUCTION.C

NAME
SETUPMON - Generates menu messages.

SYNOPSIS
SETUPMON ()

DESCRIPTION
SETUPMON, generates all the menus necessary for the tutorial.

NAME
LEARN1 - Learns function keys to mean something other than original meanings.

SYNOPSIS
LEARN1 ()

DESCRIPTION
LEARN1, defines function keys to mean specific things other than their original meanings.

NAME
MESS - Controls monitor message display.

SYNOPSIS
MESS (i)
i - Indicateswhich message to display in menu area.

DESCRIPTION
MESS, scrolls the monitor to the proper menu for display.

3.4 Query Processor Component

DATABASE.C

NAME
MAIN - Generates the first page of the database access portion of TACSYM.

SYNOPSIS
MAIN ()

DESCRIPTION
MAIN, generates the first page of the database access portion of TACSYM. Definitions of the access methods available are provided. The routine COMM is executed to control the menu driven database options. Upon return the tutorial portion of TACSYM is called.

NAME
COMM - Command menu which provides access and manipulation of the database.

SYNOPSIS
COMM (flg)
flg - Flag indicates whether to erase the work area of the monitor or not.
flg = 1 then erase; flg = 0 do not erase.

DESCRIPTION
COMM, generates the first menu which provides the user with access and manipulation database options. Upon user selection the main routine for the option selected is called. The options available are scan, insert, delete, find, and autoprint.

NAME
INSERT - Insert menu is generated to provide insertion options.

SYNOPSIS
INSERT ()

DESCRIPTION
INSERT, generates the insert menu which provides the options of inserting symbols, concepts, sources categories and primitives. The appropriate routine is called upon user selection.

DATABASE.C

NAME
DELETE - Delete menu is generated to provide deletion options.

SYNOPSIS
DELETE ()

DESCRIPTION

DELETE generates the delete menu which provides the option of deleting symbols, concepts, sources, categories and primitives. The appropriate routine is called upon user selection.

NAME
INSERT SYMBOL - Control routine for inserting a symbol into the database.

SYNOPSIS
INSERT SYMBOL ()

DESCRIPTION
INSERT SYMBOL, generates the appropriate symbol header if needed. It asks for the number of the symbol to be entered. It takes the entered number and creates a file name for it, checks if it already exists and if so asks the user if they wish to recreate it. If it is recreated then DELSYM is executed. If the symbol is being recreated or being entered for the first time, then the graphic and text file for it are created and opened. The symbol number is appended to the end of the symbol control file. The graphics window is drawn and the procedures which enter in the text information are called. The PLAY function is executed, then the remarks and flag information is asked for. Finally, the graphic and text files are closed.

DATABASE.C

NAME

INSERT PRIMITIVES - Control routine for inserting a primitive into the database.

SYNOPSIS

INSERT PRIMITIVE ()

DESCRIPTION

INSERT PRIMITIVE, generates the appropriate primitive header if needed. It asks for the name of the primitive to be entered. It creates a file name from the entered name. Checks if the file already exists and if so asks the user if they wish to recreate it. If the primitive is being recreated or being entered for the first time then the graphic file is created for it. If it's being created for the first time, its name is appended to the end of the primitive control file. The graphics window is then drawn and the PLAY function is executed to generate the primitive. Finally, the graphics file is closed.

NAME

INDEL - Control routine for inserting or deleting a concept, source or category.

SYNOPSIS

INDEL (a, b, p, m, n)

a - Flag indicating insertion or deletion mode.
1 = insert, 0 = delete

b - Flag indicating proper header to use.
1 = source, 2 = category, 3 = concept

p - Name of the control file for a source, category or concept.

m - Name of the directory for a source, category or concept.

n - Name "source," "category," or "concept", to be used in menu's

DESCRIPTION

INDEL, generates the appropriate menu for inserting or deleting a concept, source, or category. It executes SCANLIST which returns a number corresponding to an item in a list to delete or a name of an item to insert or exit. If the mode is deletion, then the reply is verified to be sure its a number. If the mode is insertion a filename is created from the returned reply. If the file already exists an error message is displayed else the concept, source or category is inserted via INSCATCONSOU. If the mode is deletion then the number returned is used as an index into the file to retrieve the name it corresponds to. A filename is created from this name. If the file doesn't exist then an error message is displayed. If the file exists and is empty then it is deleted via the DELCATCONSOU routine. If it's not empty then an error message is displayed.

DATABASE.C

NAME
DELCATCONSOU - Deletes category, concept or source from the database.

SYNOPSIS
DELCATCONSOU (p, reply, x)
p - Name of the contol file for a category, concept, or source.
reply - Name of the file in the control file to be deleted.
x - Index of the name in the file.

DESCRIPTION
DELCATCONSOU, deletes the file that was contained in the category, concept or source directory. It creates a temporary file so it can copy to it all the file names from the control file except the one to be deleted. The control file is recreated and the temporary file is written to it.

NAME
DELSYM - Asks for the symbol number to delete and create a file name for the number entered.

SYNOPSIS
DELSYM ()

DESCRIPTION
DELSYM, gets the symbol number to be deleted and creates a file name for it, then calls DELSYM1.

NAME
DELSYM1 - Deletes a symbol and all cross references for the database.

SYNOPSIS
DELSYM1 (id)
id - File name of symbol to be deleted.

DESCRIPTION
DELSYM1, opens the text file for the symbol to be deleted. From the symbol text file it gets the category, concept, source and flagged information about the symbol. For each attribute it executes DELSYMCCSF to remove its cross references. It then removes the graphc and text file for the symbol and finally removes the symbol number from the symbol control file.

DATABASE.C

NAME

DELSYMCCSF - Deletes a symbol number from its corresponding category concept source and if relevent for the flagged symbol file.

SYNOPSIS

DELSYMCCST (p, buf, id)

p - Name of the category, concept or source directory. For a flagged symbol its the control file for flagged symbols.

buf - Name of file in a specific category. For a flagged symbol buf is empty.

id - Symbol number to delete.

DESCRIPTION

DELSYMCCSF, creates file name for the category, concept or source and creates a temporary file. The contents of the entire file except the symbol number to be deleted is copied to the temporary file. The file is recreated and the temporary file is copied back to it.

NAME

SOULINE - Gets sources from the test file and calls DELSYMCCSF to delete the symbol number contained in these sources.

SYNOPSIS

SOULINE (buf, x, j, id)

buf - String from text file containing 1 or more sources.

x - Length of the buf string.

j - Index of buf where processing starts.

id - Symbol number.

DESCRIPTION

SOULINE, takes one source name at a time from buf and executes DELSYMCCSF to delete the symbol number from the corresponding source file.

DATABASE.C

NAME
DELPRIM - Deletes primitives from the database.

SYNOPSIS
DELPRIM ()

DESCRIPTION
DELPRIM, generates the appropriate primitive header and executes scanlist to generate all the primitive names the user has to select from. Upon selecting a primitive by its list number, DELPRIM creates a temporary file and copies all the primitive names from the primitive control file, except for the one selected, into the temporary file. The temporary file is copied back to the control file.

NAME
FINDSYMBOL - Generates a specific symbol in the graphic window.

SYNOPSIS
FINDSYMBOL ()

DESCRIPTION
FINDSYMBOL, asks for the symbol number to be generated. It converts the number to a file name. The symbol is generated along with its appropriate text information.

NAME
FINDPRIMITIVE - Generates a specific primitive in the graphics window.

SYNOPSIS
FINDPRIMITIVE ()

DESCRIPTION
FINDPRIMITIVE, generates a list of primitive names for the user to pick from. The primitive selected is then generated in the graphics window.

DATABASE.C

NAME
MENU - Displays the desired options a user has in accessing and manipulating the database.

SYNOPSIS
MENU (Choice)
Choice - Corresponds to the menu needed for a specific operation.

DESCRIPTION
MENU, generates a list of options that are provided with a given instruction.

NAME
MENU1 - Displays different instructions depending on current mode of operation.

SYNOPSIS
MENU1 (a, p)
a - Corresponds to the menu needed for a specific operation.
p - Character string to be used in the menu generated.

DESCRIPTION
MENU1, displays a menu giving specific instruction of what's expected and available from the keyboard.

NAME
HEADING - Generates a header on the first line of the workspace.

SYNOPSIS
HEADING (p)
p - Corresponds to the appropriate heading that is needed.

DESCRIPTION
HEADING, generates a heading in the first line of the work space that corresponds to the indicator it receives as an argument.

DATABASE.C

NAME
PLAY - Controls all procedures that perform function key operations related to "PLAY."

SYNOPSIS
PLAY (fildes)
fildes - File description of graphic file being created. It can be used for either symbols or primitives.

DESCRIPTION
PLAY, controls the execution of the function key commands and calls the appropriate routines to perform cursor movement, graphic operations, and other special functions.

NAME
EGRID - Erases the grid numbers placed outside the graphics window.

SYNOPSIS
EGRID ()

DESCRIPTION
EGRID erases the grid numbers placed around the graphics window.

NAME
REGCHAR - Controls display of text in the graphics window.

SYNOPSIS
REGCHAR (fildes, X1, Y1)
fildes - File descriptor of graphics file where text is stored.
X1, Y1 - Cursor position to display character.

DESCRIPTION
REGCHAR, displays characters from the keyboard in the graphics window. The display is stopped when either any function key is pressed or the text is at the right most position in the graphic window.

DATABASE.C

NAME
ERASE-WRITE - Executes WRITEPTS for all defined points.

SYNOPSIS
ERASE-WRITE (point, pts, chr, fd)
point - Number of defined points.
pts - Array of defined points.
chr - Character identifier reprsenting a graphic element.
fd - File descriptor of graphics file.

DESCRIPTION
ERASE-Write, executes the WRITEPTS routine to write a record of information to the graphic file for a primitive or symbol.

NAME
WRITEPTS - Converts integers to ascii and stores them in the graphics file.

SYNOPSIS
WRITEPTS (fd, a, b)
fd - File descriptor of graphics file
a,b - Coordinates of a point to be written to the graphics file.

DESCRIPTION
WRITEPTS, writes a coordinate point (a,b) to the graphics file. The points are separated by blanks in the file.

NAME
MAKEROOM - Scrolls the workspace to make room for a new graphics window at the top of the workspace.

SYNOPSIS
MAKEROOM ()

DESCRIPTION
MAKEROOM, creates room for a new graphics window at the top of the monitor area.

DATABASE.C

NAME
SYMNUMINSERT - Inserts the symbol number in the symbol text file.

SYNOPSIS
SYMNUMINSERT (fdt, sid)
fdt - File descriptor for text file.
sid - Symbol number.

DESCRIPTION
SYMNUMINSERT, writes the symbol number in the symbol text file. A "N" is placed before the number as a number indicator.

NAME
SYMCATINSERT - Gets and stores the category name for a symbol.

SYNOPSIS
SYMCATINSERT (fdt, id)
fdt - File descriptor for text file.
id - Symbol number.

DESCRIPTION
SYMCATINSERT, gets the category name via kayboard. It appends a " " to the name and writes the name to the graphic text file. It's preceded by "K:" which indicates category name. If a file for the category name in the category directory doesn't exist then one is created by executing INSCATCONSOU. The file is opened and the symbol number is appended to it.

DATABASE.C

NAME

SYMCONINSERT - Gets and stores primary and secondary concepts for a symbol.

SYNOPSIS

SYMCONINSERT (fdt, d)
fdt - File descriptor for a text file.
id - Symbol number.

DESCRIPTION

SYMCONINSERT, get the primary concept via keyboard. It appends a " " to the name and writes it to the graphics text file. Its preceded by a "C" which indicates concept name. If a file for the concept name in the concept directory doesn't exist then one is created by executing INSCATCONSOU. This file is opened and the symbol number is appended to it. TACSYM allows the users two more lines for the entry of secondary concepts. These are entered into the graphics text file.

NAME

SYMSORINSERT - Gets and stores the sources of the symbols.

SYNOPSIS

SYMSORINSERT (fdt, id)
fdt - File descriptor for text file.
id - Symbol number.

DESCRIPTION

SYMSORINSERT, get the source via keyboard. Sources are prompted until the user exits the source input mode or the space provided (2 lines) for sources is filled. The sources are written to the text file. The first line of sources is preceded by "S:" which indicates source names. If a file for the source name in the source directory does not exist then one is created by executing INSCATCONSOU. This file is opened and the symbol number is appended to it.

DATABASE.C

NAME
INSCATCONSOU - Creates a category, concept or source within the database.

SYNOPSIS
INSCATCONSOU (p, buff)
p - Control file name for category, concept or source.
buff - Name of category, concept or source.

DESCRIPTION
INSCATCONSOU, creates a file within the proper category, concept or source directory. The file name corresponds to the name of the category, concept or source. It also appends the file name to the control file.

NAME
SYMREMINSERT - Gets and stores remarks and flags information for a symbol.

SYNOPSIS
SYMREAMINSERT (fdt, id)
fdt - File descriptor for text file.
id - Symbol number.

DESCRIPTION
SYMREMINSERT, provides the users with two lines to insert any remarks about the symbol they just created. The first line is preceded by a "R:" in the text file. The user is then asked if the symbol should be flagged. If yes is answered then a "Y" is written to the text file, otherwise a "N" is written. The flag is preceded by a "F:" to indicate flag information. If the syumbol is flagged, then the control file for flagged symbols is opened and the symbol number is appended to it.

NAME
LEARN - Learns specific keys on the keyboard to be equivalent to other meanings rather than their original.

SYNOPSIS
LEARN ()

DESCRIPTION
LEARN, defines keys on the keyboard to equal specific meanings that are used in the code.

DATABASE.C

NAME
RELEARN - Learns all TACSYM function keys to mean the same thing.

SYNOPSIS
RELEARN ()

DESCRIPTION
RELEARN, defines all the TACSYM keys to be the same.

NAME
DRAWBOX - Sets the graphics window and executes a box. (A box provides the area in which symbols may be drawn.)

SYNOPSIS
DRAWBOX (flag)
flag - Flag to indicate whether to call up a box.
Flag = -1 don't draw Flag = 1 then draw.

DESCRIPTION
DRAWBOX, defines the graphics window and if flag is not equal -1 executes box to draw a box around the graphics window.

NAME
ADFIG - Adds a symbol or primitive to the graphics file.

SYNOPSIS
ADFIG (fildes, x, y, symflag)
fildes - File description of graphics file.
x,y - Current cursor location in graphics window.
symflag - Flag indicating if adding a symbol or primitive.
symflag = 1 symbol symflag = 0 primitive

DESCRIPTION
ADFIG, gets the symbol number or primitive name that the user wishes to include in the graphics window. A symbol or primitive identifier (S or P) is written in the graphics file. The symbol or primitive file name follows. The (x,y) relative position is put after the file name.

DATABASE.C

NAME
REDO - Deletes the last component that the user inserted in the graphic file.

SYNOPSIS
REDO (fildes)
fildes - File descriptor of graphic files.

DESCRIPTION
REDO, takes the current graphics file, whether it be a symbol or a primitive and eliminates the last graphical component added to it. It erases the graphics window and redraws the graphics in it except the last item added.

NAME
REDRAW - Redraws the graphics window contents.

SYNOPSIS
REDRAW (fildes)
fildes - File descriptor of graphics file.

DESCRIPTION
REDRAW, erases the entire graphics window and redraws its contents.

NAME
TEXTGEN - Generates the text that corresponds to a graphic symbol.

SYNOPSIS
TEXTGEN (id, line)
id - Symbol number.
line - Line on screen to start positioning.

DESCRIPTION
TEXTGEN, opens the text file that cooresponds to a graphic symbol. It dumps the text informtion to specified areas on the screen.

DATABASE.C

NAME
LOADBUF - Loads a buffer with text until a new line character is encountered.

SYNOPSIS
LOADBUF (c, buf, fd)
c - Character which starts buffer.
buf - Buffer to load information,.
fd - File descriptor of text file.

DESCRIPTION
LOADBUF, dumps text from the graphics symbol text file into a buffer. A newline delimits the buffer.

NAME
INSERZEROS - Prepends zeros to the three numbers that comprise a symbol number.

SYNOPSIS
INSERZEROS ()

DESCRIPTION
INSERZEROS prepends zeros to three numbers that comprise a symbol number. The zeros are added if the numbers digits do not equal their maximum size (i.e. for category 1 the number will be 01).

NAME
WCNT - Breaks up keyboard input using special characters as delimiters.

SYNOPSIS
WCNT (buff)
buff - Keyboard input buffer.

DESCRIPTION
WCNT, scans the buffer for special characters. Once a special character is encountered a new word is created from all the characters upto this special character. A maximum of three words is allowed to be created. It returns the number of words created.

DATABASE.C

NAME

CREATFILNAME - Creates a file name from previously broken down names.

SYNOPSIS

CREATFILNAME (buff, n)

buff - Name to create.
n = Number of words in the buffer.

DESCRIPTION

CREATFILNAME, creates a file name from the number of words broken up from the input name. The file name could look like the following:

for 1 word	6 characters, a period and three characters or, 6 or less characters and a period.
for 2 words	6 or less characters, a period and three or less characters.
for 3 words	6 or less characters (made by words 1 and 2) a period and 3 or less characters.

SCAN.C

NAME
SCAN - Controls all database scanning mechanisms.

SYNOPSIS
SCAN ()

DESCRIPTION
SCAN, controls all database scanning. Initially it generates a definition of its menu contents. Upon the users selection it calls the appropriate procedures to access the database. SCAN matches the users reply with the proper procedure and arguments to carry out the commands.

NAME
SCANLIST - Generates page(s) of sources, concepts, categories and primitive names.

SYNOPSIS
SCANLIST (p, reply, reply1)
p - Control filename corresponding to the appropriate database attribute.
reply - Integer value cooresponding to proper heading in "listheading" routine.
reply1 - Users returned reply.

DESCRIPTION
SCANLIST, generates an enumerated list of sources, concepts, categories or primitives. These lists are used to help the user in selection and verification. A paging mechanism is also used in the event of large lists.

SCAN.C

NAME

SCANLATER - Scans the database by the appropriate selection made in scanning a source, concept or category list.

SYNOPSIS

SCANLATER (p, reply, p2)

p - File which contains the name of the specific source, concept or category to scan by.

reply - Integer character corresponding to the entry in the file where namee is located.

p2 - Directory name where the specific file selected from the list is located.

DESCRIPTION

SCANLATER, converts the integer ASCII character "reply" to integer and uses it as an index to get the record in the file that was previously selected. This record name is converted to a filename and appended to its directory. The entire pathname is sent to SCANSYMS.

NAME

SCANPRIMS - Generates consecutive pages of primitives contained in in the database.

SYNOPSIS

SCANPRIMS (p)

p - Name of control file which contains primitive names.

DESCRIPTION

SCANPRIMS, opens the primitive control file which contains the names of all the primitives. It converts these names into file names and opens the files. It generates five primitives per page and displays a message asking the users if they wish for another page or want to exit.

SCAN.C

NAME
SCANSYMS - Generates consecutive pages of symbols contained in the database.

SYNOPSIS
SCANSYMS (p)
p - Name of control file which contains symbol names.

DESCRIPTION
SCANSYMS, opens the symbol control file which contains the numbers of all the symbols. These numbers are converted to symbol graphic and text file names and are generated on the screen. SCANSYM generates symbols for scanning and for AUTOPRINT. Scanning generaters 5 symbols per page while autoprint generates 9.

NAME
LISTHEADING - Generates the proper heading for scanning routines.

SYNOPSIS
LISTHEADING (type, n)
type - Corresponds to which heading is desired.
n - Used to calculate columns.

DESCRIPTION
LISTHEADING, Generates proper heading for scan routine.

NAME
PRINTSYMS - Gets page capacity and calls SCANSYMS.

SYNOPSIS
PRINTSYMS ()

DESCRIPTION
PRINTSYMS, used to call SCANSYMS for AUTOPRINT option.

SCAN.C

NAME
SCANMENU - Generates scan menu.

SYNOPSIS
SCANMENU (i)
i - Menu to generate

DESCRIPTION
SCANMENU, generates the scan menu.

3.5 Software Support System

ARC1.C

NAME
ARC - Generates an ARC in the graphics window.

SYNOPSIS
ARC (point, pts, nfill)
point - Interger number of defined points.
pts - Interger array of defined points.
nfill - Interger flag indicating whether the arc is to be filled or not.
1 = Fill 0 = Don't fill

DESCRIPTION
ARC, processes three points and generates an arc in the graphics window The first and third points define the edges of the arc and the second point defines the distance from the arc's base (defined as a line connecting points one and three) to its peak. NFILL is a flag used to determine if the arc should be filled or not.

ARC2.C

NAME
COMPARC - Computes points on the arc.

SYNOPSIS
COMPARC (tar)
tar - Integer array of points used to calculate new points on the arc.

DESCRIPTION
COMPARC, is called by the arc routine. It computes the points that lie on the arc. It is called repeatedly from arc until all necessary points to generate the appropriate arc are computed.

GRAPHIC.C

NAME
BOX - Generates a perimeter box for the graphics window.

SYNOPSIS
BOX ()

DESCRIPTION
BOX, displays a box around the perimeter of the graphics window.

NAME
CALL-FUNC - Calls a specific graphic function.

SYNOPSIS
CALL-FUNC (points, n, c)
points - Array of defined points.
n - Number of defined points.
c - Character which indicates which graphic routine to call.

DESCRIPTION
CALL-FUNC, calls the circle, arc, connectpt, or fillpoly function depending on the "c" argument.

NAME
CIRCLE - Generates a circle in the graphics window.

SYNOPSIS
CIRCLE (point, pts, fill)
point - Number of defined points.
pts - Array of defined points.
fill - Flag indicating whether to fill circle or not.
Fill = 1 fill circle Fill = 0 don't fill circle

DESCRIPTION
CIRCLE, generates a circle in the graphics window. It requires two points to define the circle's diameter. If the "fill" flag = 1 then the circle is filled.

GRAPHIC.C

NAME
CLIP - Clips vectors that pass through the graphics window's edge.

SYNOPSIS
CLIP (X1, Y1, X2, Y2)
X1, Y1 - Coordinates of first point which defines a vector.
X2, Y2 - Coordinates of second point which defines a vector.

DESCRIPTION
CLIP, checks if a vector defined by two points passes through the graphics window. If it does, new points are calculated to equal the points where it passes through the window. A 1 is returned if the resulting points lie in the window.

NAME
CODE - Assign binary value used in CLIP routine depending on X,Y location.

SYNOPSIS
CODE (x,y)
x,y - One point defining a vector.

DESCRIPTION
CODE, assigns a binary value which is used in CLIP. The value is assigned according to the x and y locations.

NAME
CONNECTPTS - Connects defined points.

SYNOPSIS
CONNECTPTS (point, pts, flflag)
point - number of defined points.
pts - Array of defined points.
flflag - flag indicating whether to connect the first and last point defined.
flflag = 1 connect first and last point
flflag = 0 don't connect first and last point.

DESCRIPTION
CONNECTPTS, generates vectors connecting 2 or more defined points. These points are connected in the order of definition. If the flflag = 1 then the first and last point defined is connected by a vector. These points are not connected if flflag = 0.

GRAPHIC.C

NAME
CURSOR - Executes movement of the cursor in the graphics window.

SYNOPSIS
CURSOR (xch, ych, x,y)
xch - Desired change in x movement.
ych - Desired change in y movement.
x,y - Current location of cursor.

DESCRIPTION
CURSOR, erases the current cursor in the graphics window and checks if the desired new cursor position falls in the graphics window. If the new cursor will not fall in the window then the cursor is redrawn at its old position and a 0 is returned. If it does fall in the window then the cursor is drawn at it's new position and a 1 is returned.

NAME
DCROSS - Displays representation for a defined point.

SYNOPSIS
DCROSS (x,y)
x,y - Point to generate defined point representation.

DESCRIPTION
DCROSS, generates an 'X' at the x,y location of a defined point.

NAME
DPLUS - Generates cursor representation.

SYNOPSIS
DPLUS (x,y)
x,y - Point to generate cursor representation.

DESCRIPTION
DPLUS, generates a 't' at the x,y location of the current beam position.

GRAPHIC.C

NAME
DUMP - Generates a primitive or symbol relative to the current cursor position.

SYNOPSIS
DUMP (fildes, x,y)
fildes - File descriptor of symbol or primitive.
x,y - Coordinates to offset the primitive or symbol.

DESCRIPTION
DUMP, gener ates a primitive or symobl relative to the x,y offset it receives as an argument. It can be called recursively to generate embedded symbols and primitives.

NAME
ERASEPTS - Erases all defined points from the graphics window.

SYNOPSIS
ERASEPTS (point, pts)
point - Number of defined points.
pts - Array of defined points.

DESCRIPTION
ERASEPTS, erases all defined points (represented by 'X') from the graphics window.

NAME
FILPOLY - Fills a simple structured polygon with 3 to 4 sides.

SYNOPSIS
FILPOLY (pts, point)
pts - Array of defined points.
point - Number of defined points.

DESCRIPTION
FILPOLY, generates a filled polygon. The polygon can have 3 or 4 sides, where the top and bottom must be parallel. In the case of the triangle (3 sides) one side must be perfectly horizontal.

GRAPHIC.C

NAME
GENERATE - Sends DUMP routine the relative position to generate a primitive or symbol.

SYNOPSIS
GENERATE (id, x,y)
id - Name of file to generate.
x,y - Relative location to position files contents.

DESCRIPTION
GENERATE, opens the file that is to be generated. It sends the file descriptor and the x,y relative location to put the file, to DUMP.

NAME
GRID - Overlays a grid on the graphics window.

SYNOPSIS
GRID ()

DESCRIPTION
GRID, generates a grid on the graphics window. This is used to help in cursor positioning in creating graphics.

NAME
REDOPTS - Redraws defined points.

SYNOPSIS
REDOPTS (point, pts)
point - Number of defined points.
pts - Array of defined points.

DESCRIPTION
REDOPTS, redraws an 'X' over all the defined points in the graphics window.

RAWCOOK.C

NAME
RAW - Sets Tektronix terminal not to echo text onto the monitor.

SYNOPSIS
RAW ()

DESCRIPTION
RAW, sets the Tektronix terminal so no text is echoed on the monitor from the keyboard or the host.

NAME
COOK - Sets the Tektronix terminal to echo text onto the monitor.

SYNOPSIS
COOK ()

DESCRIPTION
COOK, sets the Tektronix terminal to echo text on the monitor sent from the keyboard or the host.

STR.C

NAME
CATS - Concatinate two strings into a resulting string.

SYNOPSIS
CATS (str1, str2, result)
str1, str2 - Two strings that are to be concatenated.
result - String that two strings result is stored.

DESCRIPTION
CATS, concatenates two strings into "result."

NAME
CITOA - Converts integer to character string.

SYNOPSIS
CITOA (i)
i - Integer to convert to character string.

DESCRIPTION
CITOA, converts an integer to a character string by executing ITOS.

NAME
COMPS - Two strings are compared for equality.

SYNOPSIS
COMPS (str1, str2)
str1,str2 - Two strings to be compared.

DESCRIPTION
COMPS, compares two strings and returns 1 if they are identical, otherwise it returns 0.

STR.C

NAME
COPYS - Creates a copy of a string.

SYNOPSIS
COPYS (from, to)
from - String being copied.
to - New string.

DESCRIPTION
COPYS, assigns each character of the string "from" to the string "to."

NAME
FINDI - Sequential search for an integer in an array of integers.

SYNOPSIS
FINDI (array, length, integer)
array - Array of integer.
length - Number of integers in the array to sequentally search.
Integer - Integer to find in the array.

DESCRIPTION
FINDI, sequentally searches an array of integers for a specific integer and returns the indice of the array where the number was found. Otherwise it returns -1.

NAME
FLUSHL - Flushes all characters for the terminal's buffer.

SYNOPSIS
FLUSH ()

DESCRIPTION
FLUSHL, flushes all characters from the terminal's buffer until a carriage return is encountered.

STR.C

NAME
FLUSHLP - Flushes all the characters from the terminal's buffer; it checks for 1 character terminal input.

SYNOPSIS
FLUSHLP ()

DESCRIPTION
FLUSHLP, flushes all characters from the terminal's buffer until a carriage return is encountered. It returns a 0 if one character and a carriage return was entered, otherwise a 1 is returned.

NAME
GETCH - Reads a character from a file.

SYNOPSIS
GETCH (fildes)
fildes - File descriptor.

DESCRIPTION
GETCH, reads one character from an input file and returns that character.

NAME
GETCOOR - Reads a pair of coordinates needed to generate a symbol or primitive.

SYNOPSIS
GETCOOR (file, x,y, debug)
file - File to get coorinator from.
x,y, - Variable to return coordinate in.
debug - File descriptor for debugging an error in the database i.e. file doesn't exist.

DESCRIPTION
GETCOOR, opens a file and gets the first x,y coordinate pair of the first record in the file. If the file is not found then an error message is written to the debug file.

STR.C

NAME
GETNUM - Gets an ascii character number and returns it as an integer.

SYNOPSIS
GETNUM (fd)
fd - File descriptor of file to get number from.

DESCRIPTION
GETNUM, reads a file until it encounters an ascii character number and converts it to integer. GETNUM returns the integer.

NAME
GETRECORD - Gets a record of data from a previously loaded buffer.

SYNOPSIS
GETRECORD (fd, buf, rec, n)
fd - File descriptor of file of data.
buf - Buffered area to store block reads.
rec - Record area to store a record read from buf.
Record defined as data terminated by a newline.
n - Maximum number of bytes to read into buffer.

DESCRIPTION
GETRECORD, copies contigious bytes from the buffer to the record until a newline is encountered. Once the entire buffer is empty and EOF has not been reached then the buffer is filled again with bytes from the file. A 1 is returned on a successful copy to the record and a 0 is returned upon EOF or read error.

NAME
GETS - Gets a string from a file.

SYNOPSIS
GETS (fildes, str)
fildes - File descriptor.
str - Store that is gotten from file.

DESCRIPTION
GETS, reads a character string from a file and assigns it to STR.

STR.C

NAME
GETFLS - Gets a fixed length string from a file.

SYNOPSIS
GETFLS (fildes, str, cnt)
fildes - File descriptor of file.
str - String to get.
cnt - Maximum length string can be.

DESCRIPTION
GETFLS, gets a string from a file which can be no longer than "cnt."

NAME
GET_STR - Gets a string constrained by a delimiter.

SYNOPSIS
GET_STR (fd, s, d)
fd - File descriptor.
s - String to get.
d - Delimiter to indicate end of string.

DESCRIPTION
GET_STR, gets a string from a file. The string is read until a newline character or the delimiter is encountered.

NAME
ITOS - Converts integer to a string.

SYNOPSIS
ITOS (str, i)
str - String that is created from integer.
i - Integer to convert to string.

DESCRIPTION
ITOS, converts an integer number to a character string. It returns the string and its length.

STR.C

NAME
LENS - Calculates the length of a string.

SYNOPSIS
LENS (str)
str - String itself.

DESCRIPTION
LENS, computes the length of a character string and returns that length.

NAME
MATCH - Searches a character string for a character.

SYNOPSIS
MATCH (isthischar, inthisstring)
isthischar - Character being searched.
inthisstring - String being searched.

DESCRIPTION
MATCH, searches a character string for a specific character and returns a 1 if that character is found and a 0 otherwise.

NAME
PUTD - Takes an integer, converts it to ascii and writes it to a file.

SYNOPSIS
PUTD (fd, integer)
fd - File descriptor of file to put converted integer.

DESCRIPTION
PUTD, converts an integer to an ascii character string and writes the string to a file.

STR.C

NAME
PUTS - Writes a string to a file.

SYNOPSIS
PUTS (fildes, str)
fildes - File descriptor.
str - String to write to file.

DESCRIPTION
PUTS, writes a string to a file.

NAME
SEARCH - Searches character array for a string.

SYNOPSIS
SEARCH (array, abytes, string, sbytes)
array - Character array.
abytes - Number of bytes in array
string - Character string.
sbytes - Number of bytes in string.

DESCRIPTION
SEARCH, searchs a character array for a character string. If found the index where the string starts in the array is returned else a -1 is returned.

TEK1.C

NAME
BEAM - Positions the beam position in the Tektronix graphics window at some point.

SYNOPSIS
BEAM (x,y)
x,y - Integer values of coordinates to position beam at.

DESCRIPTION
Beam, checks if the coordinates it receives falls in the graphics window. If they do then it positions the beam there and returns 1, otherwise it returns 0.

NAME
CLEAR - The keyboard and function keys are set to their original meanings

SYNOPSIS
CLEAR ()

DESCRIPTION
CLEAR, sets all keyboard and function keys, that may have been redefined to mean something else back to their original meanings.

NAME
DLINE - Deletes lines from the screen.

SYNOPSIS
DLINE (a)
a - Integer number of lines to be deleted.

DESCRIPTION
DLINE, deletes lines from the screen, relative to the cursor position. The number of lines to delete is determined by its parameter.

TEK1.C

NAME
DOWN - Move cursor down a page.

SYNOPSIS
DOWN (a)
a - Integer number of lines to move cursor down.

DESCRIPTION
DOWN, scrolls the cursor down the Tektronix's screen. The number of lines to move the cursor is determined by its parameter.

NAME
ENHANCE - The enhanced or standard attribute is attached to the current cursor position.

SYNOPSIS
ENHANNCE (ch)
ch - Character indicating if standard or enhanced visual attributes should be displayed.

DESCRIPTION
ENHANCE, receives a single character "s" or "e" which visually displays the standard attribute or the enhanced attribute respectively.

NAME
ERASE - Complete easure of the defined workspace, monitor, or graphics area.

SYNOPSIS
ERASE (ch)
ch - Character indicating what area of the monitor to erase.

DESCRIPTION
ERASE, receives a character w, m, or g which erases the workspace, monitor, or graphics area respectively.

TEK1.C

NAME
GRAPHIC - Defines the graphics area of the monitor.

SYNOPSIS
GRAPHIC (a,b,c,d)
a,b,c,d - Integer number indicating the first row, last row, first column, and last column of graphics window, respectively.

DESCRIPTION
GRAPHIC, receives the rows and columns to be set as the graphics window.

NAME
HCOPY - Generates hard copy of workspace, monitor or the entire screen.

SYNOPSIS
HCOPY (ch)
ch - Character indicating what area of the screen to generate the hard copy.

DESCRIPTION
HCOPY, receives a character w,m, or s indicating whether to print a hardcopy of the workspace, monitor area, or the entire screen, respectively.

NAME
ILINE - Inserts lines on monitor.

SYNOPSIS
ILINE (a)
a - Integer number of how many lines to insert.

DESCRIPTION
ILINE, receives an integer indicating how many lines to insert into the monitor. It inserts that many lines.

TEK1.C

NAME
JUMP - Quick relocation of cursor in the workspace.

SYNOPSIS
JUMP (x,y)
x,y - Integer line, column position in the workspace.

DESCRIPTION
JUMP, will move the cursor to the line, column position in the workspace which corresponds to its parameters.

NAME
LEFT - Moves cursor to the left.

SYNOPSIS
LEFT (a)
a - Integer number of how many positions to move the cursor to the left.

DESCRIPTION
LEFT, moves the cursor from its current position to the left. The number of positions it is moved is determined by its parameter.

NAME
LINE - Sets the type of line to draw via the vector command.

SYNOPSIS
LINE (ch)
ch - Character which determines type of line to set.

DESCRIPTION
LINE, sets the different line representation available. Lines range from dots to solid. Blank lines are also available for erasing purposes.

TEK1.C

NAME
MONITOR - Sets the monitor area and its input constraints.

SYNOPSIS
MONITOR (str)
str - String which can indicate the monitor area, or what can communicate to the monitor (the host or keyboard).

DESCRIPTION
MONITOR, receives a character string which can be the number of lines for the monitor, or the character h which directs text from the host to the monitor, or the character K which directs text from the keyboard to the monitor.

NAME
RIGHT - Moves cursor to the right.

SYNOPSIS
RIGHT (a)
a - Integer number of how many positions to move the cursor to the right.

DESCRIPTION
RIGHT, moves the cursor from its current position to the right. The number of positions it is moved is determined by its parameter.

NAME
TEKCHAR - Generates character in the graphics window.

SYNOPSIS
TEKCHAR (c)
c - Character to be displayed in the graphics window.

DESCRIPTION
TEKCHAR, receives a character which is sent to the graphics window and displayed at the current beam position.

TEK1.C

NAME
TEKSTR - Generates string in graphics window.

SYNOPSIS
TEKSTR (str)
str - String to be displayed in the graphics window.

DESCRIPTION
TEKSTR, receives a string of text which is displayed in the graphics window at the current beam position.

NAME
UP - Move cursor up.

SYNOPSIS
UP (a)
a - Integer number to move the cursor up the screen.

DESCRIPTION
UP, moves the cursor up the screen. The number of lines it moves the cursor is determined by its parameter.

NAME
VECTOR - Draws line between two points.

SYNOPSIS
VECTOR (X1,Y1,X2,Y2)
X1,Y1,X2,Y2 - Integer coordinates which represent points one and two.

DESCRIPTION
VECTOR, draws a line between two points it receives. The type of line it draws is determined by the line command.

TEK1.C

NAME

WORKSPACE - Sets the workspace area and its input constraints.

SYNOPSIS

WORKSPACE (str)

str - String which indicates the workspace area, or what can communicate to the workspace (the host or keyboard).

DESCRIPTION

WORKSPACE, receives a character string which can be the number of lines for the workspace, or the character h which directs text from the host to the workspace, or the character K which directs text from the keyboard to the workspace.

TRIG.C

NAME
ATAN - Computes arc tangent of a slope.

SYNOPSIS
ATAN (x)
x - Double precision variable of the slope of a line.

DESCRIPTION
ATAN, computes the inverse tangent of the slope X and returns the answer in radian measure.

NAME
SQRT - Computes square root of 1/Y.

SYNOPSIS
SQRT (y)
y - Double precision variable.

DESCRIPTION
SQRT, computes the square root of 1 divided by y. This routine is used exclusively for ATAN. Returns the answer in double precision.

NAME
POW - Raises some number to some power.

SYNOPSIS
POW (y,n)
y,n - Double precision variable.

DESCRIPTION
POW, computes y raised to the nth power. It returns the answer in double precision.

TRIG.C

NAME
SIN - Computes the sine of some number.

SYNOPSIS
SIN (a)
a - Double precision variable

DESCRIPTION
SIN, receives an angle in radian measure and returns the sine of that angle. The sine is returned in double precision radian measure.

NAME
COS - Computes the cosine of some number.

SYNOPSIS
COS (a)
a - Double precision variable.

DESCRIPTION
COS, receives an angle in radian measure and returns the cosine of that angle. The cosine is returned in radian measure.

NAME
FABS - Computes the absolute value of double precision variable.

SYNOPSIS
FABS (z)
z - Double precision variable.

DESCRIPTION
FABS, computes the floating point absolute value of a variable and returns it as double precision.

4. DETAILED OPERATIONAL DESCRIPTION

4.1 Initialization of System

Assuming the user is currently logged on the system and positioned at the proper directory which contains the TACSYM program, type TACSYM. This should generate the first page of TACSYM with three options available for selection. By typing a 1, the tutorial part of TACSYM will be executed. This causes the current page to be erased and a wait message to be displayed while the tutorial menus are being generated. By typing a 2, the catalogue access and the modification part of TACSYM will be executed. This causes the current page to be erased and the first menu of the access and modification section to be displayed. By typing a 3, the program is terminated and the screen erased with the system prompt sign displayed. If the EXIT key is pressed at this point the prompt sign will show in the monitor area. Simply typing TACSYM again followed by the number 3 will properly clear the screen.

4.2 Tutorial Processor Component

Once at the first page of the tutorial, a tree diagram is seen with an indication of where the user currently is in the system. The enhanced text is a selection mechanism which will be used throughout the tutorial. It is controlled by the cursor function keys (function keys 2 and 3). The menu explains the options the user has at the current level of the system. By pressing function keys 2 and 3 the enhanced text will alternate until CONFIRM or EXIT is pressed. CONFIRM will take the user to the section of the tutorial that is highlighted while an EXIT will take the user up one level to the first page. CONFIRM acts as a selection button, taking the user one more level into the tutorial. Conversely, EXIT acts as a backing

out button, taking the user one more level out of the tutorial to the previous selection menu. The tutorial is broken up into two parts. Part one is the System Operations and part two is the Executive Summary. System Operations and Executive Summary are further broken down into three sections each.

Systems Operations

Systems operations consists of how to use the keyboard, catalogue structure description and a narrative on how to access the catalogue. How to use the keyboard is broken down into a definition and illustration walk-through of the function keys which are positioned at the top of the keyboard and a symbol and primitive creation practice session. The definition and illustration walk-through reads like a book, in that you can page forward or backward one page at a time, exit at any time, or repeat a page. The symbol and primitive creation practice session provides the user with a graphics window where he can experiment with the function keys. Upon exiting from this practice session the tutorial menus will be generated again. The catalogue structure consists of a brief explanation of its capabilities and characteristics. How to access the catalogue is a brief paragraph explaining keyboard operations of how to enter and exit levels of the system.

Executive Summary

The Executive Summary consists of an overview, general rules and sample symbols. The overview is broken down into a summary of the symbology program, results of the symbology survey, symbol discriminability and TACSYM system development. The general rules consist of development rules for specific

symbology sources for the UNIT, WEAPON, and POST and INSTALLATION symbols. It also explains color rules. Sample symbols consist of a range of symbols contained in the database.

4.3 Query Processor Component

The top menu of the symbology catalogue contains six options for the user; one is to scan the database, two is to insert something into the database, three is to delete something from the database, four is to find something in the database, five is to automatically print all symbols in the database and six is to exit the symbology catalogue. The initial title and definition on the first menu are only displayed upon first entry into the catalogue.

Scan

In scanning the database six options are available; one is scan by source, two is scan by category, three is scan by concept, four is scan all symbols, five is scan all flagged symbols and six is scan all primitives. In scanning by the first three, an enumerated list of potential items to scan by is generated. The user can enter in the appropriate number and press confirm, exit, or if more than one page of items exists, press confirm for the next page. If a number is entered then the symbols corresponding to that numbered item are displayed. In scanning all symbols, flagged symbols or primitives, TACSYM generates a page of the item selected and waits for a confirm to generate the next page, or an exit which goes back to the scan menu. The explanation of the scan capabilities at the top scan menu is only displayed on the first pass into the scanning options.

Insert/Delete

To insert or delete to and from the database, five options are available; one is symbol, two is concept, three is source, four is category and five is primitive. To insert a symbol, the number the user wishes to associate the symbol with is asked for in the form of n.n.n. If a symbol already exists with that number then the user is asked to press confirm if he wants to replace it. During symbol insertion the category name is asked for, then the primary concept and then the secondary concept. Two lines are allocated for the secondary concept. The sources are requested one at a time until the user indicates by pressing exit that he is finished entering them or no room exists for more. The symbol is then created via function keys. Two lines are then allocated for remarks and finally the user is asked if he/she wishes to flag the symbol or not. To delete a symbol, the system asks for the symbol number of the symbol associated with it. To insert a concept, source or category the system generates a list of the specified item's contents and asks the user to type in the new item not already in the list. To delete a concept, source or category the system generates a list of the specified item's contents and asks the user to enter the number corresponding to the item he/she wishes to delete. If this item is used anywhere else in the database (i.e., it's a concept of an existing symbol), it will not be removed. To insert a primitive the name is asked for. If the name of the new primitive is the same as one already in the system the user is asked to confirm if he/she wishes to recreate it. During primitive insertion the function keys are then used in creating the primitive. To delete a primitive the list of primitives is generated and the user is asked to enter the corresponding number of the primtive he/she wishes to delete. Note an already

existing primitive or symbol may be used by another primitive or symbol; be sure that this is not the case before deleting it.

Find

Find is used to generate a specific symbol or primitive. To find a symbol, enter in the symbol number in the form of n.n.n; this will display the symbol for that symbol number on the monitor. If the symbol does not exist, then an error message will appear. In finding a primitive, a list of primitive names is generated and the user is asked to enter the number corresponding to the primitives he/she wishes to display. This primitive is then displayed on the monitor.

Autoprint

Autoprint requires the hard copy device to be turned on. Autoprint will generate all the symbols currently in the database. This option allows the user to leave the system unattended while it generates and copies all symbols.

4.4 User Interface

The monitor has the capability of utilizing two areas of its screen. During the execution of TACSYM the lower portion of the monitor is used as a menu area while the rest of the monitor is used for information display. The menu areas instruct the user as to what is expected from the keyboard.

TACSYM does key validation of any number or name entered into the system. TACSYM will check if the command is correct and process it accordingly. Otherwise TACSYM will ask for the command again or display a message

indicating what was wrong. TACSYM was designed with the user in mind. Allowing the system to guide the user will be to his/her advantage.